UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Extended VR Core

Treball Final de Grau

Grau en Disseny i Desenvolupament de Videojocs

Cognoms: Gironés Bosch Nom: Miquel

Pla: 2014

Director: Zuñiga Zarate , Ana Gabriela

Índex

Resum 4

Paraules clau 5

Enllaços 5

Índex de taules 6

Índex de figures 7

Glossari 8

1. Introducció 16

1.1 Motivació 16

1.2 Formulació del problema 16

1.3 Objectius generals del TFG 17

1.4 Objectius específics del TFG 17

1.5 Abast del projecte 18

2. Estat de l'art 19

2.1 Estudi de Mercat 20

3. Gestió del projecte 21

3.1 Procediment i Eines per al seguiment del projecte 21

3.1.1 GANTT 21

3.1.2 GitHub repositori en xarxes, Git eines de control de versions 21

3.2 Eines de validació 21

3.3. DAFO 22

3.4. Riscos i pla de contingències 22

3.5. Anàlisi inicial de costos 23

4. Metodologia 24

5. Desenvolupament del projecte 27

5.1 Fase 0: Preparació del projecte. 27

5.2 Fase 1: Arquitectura de Mòduls 28

5.3 Fase 2: ModuleInput 31

Interactivitat física 35

Interactivitat per raycast o a distancia 36

Sistema de Interactive Items 37
5.4 Fase 3: ModuleEvents 41
Event 41
Sistema d'events 41
Implementació ModuleEvent 42
5.5 Fase 4: Test Scene 47
5.6 Fase 5: Controls 51
5.7 Fase 4: Testeig i correcció de bugs 55
5.7 Fase 5: Publicació del projecte 57
6. Conclusions i treballs futurs 60
7. Bibliografia 65

Resum

La realitat virtual és una tecnologia emergent i cada cop té més rellevància en la industria de la informàtica. Degut a aquest creixement, cada vegada hi ha més empreses i més desenvolupadors que treballen amb realitat virtual. El problema és que hi ha diferents dispositius amb diferents capacitats i costos y, per tant, diferents maneres de desenvolupar per a aquest ampli ventall de possibilitats. Això suposa un problema ja que provoca que les aplicacions tinguin dificultats alhora de suportar multiplataforma.

És per això que l'objectiu d'aquest projecte és crear una API o "Asset" que permet a les aplicacions tenir multiplataforma aplicant els mínims canvis. Consisteix en crear un seguit d'eines considerant diferents dispositius per tal de que el desenvolupament d'una aplicació no experimenti canvis o n'experimenti els mínims canvis per tal de ser usada en un dispositiu o en un altre. Sempre tenint en compte les capacitats dels diferents dispositius.

Paraules clau

Realitat Virtual, Core, Asset, Package, Multiplataforma, Wrapper, Asset Store.

.

Enllaços

Enllaç on està el projecte de Unity amb tot el codi disponible per descarregar:

https://github.com/SenyoraCatolica/ExtendedVRCore

Enllaç a al paquet dins de la Asset Store de Unity (Inactiu fins que s'accepti el projecte):

https://assetstore.unity.com/packages/slug/147900

Índex de taules

Taula 1: Diagrama de Gantt.....Pag. 14

Taula 2: Estructura per mòduls del EVRCore.....Pag. 21

Taula 3: Estructura de Selecció de Hardware.....Pag. 24

Taula 4: Asset InputGeneralConfig.....Pag. 25

Taula 5: Asset InputGeneralConfig amb tipus de dispositius disponibles.....Pag. 25

Taula 6: Configuració de Interactive Items des de l'editor .....Pag. 37

Taula 7: Diagrama UML del sistema de InteractiveItems.....Pag. 38

Taula 8: Exemple d'un EventArgs del tipus booleà.....Pag. 41

Taula 9: Exemple d'un EVREvent del tipus booleà.....Pag. 42

Taula 10: Exemple d'una instància d'event des de l'editor.....Pag. 42

Taula 11: Crida d'un RaiseEvent del tipus EVREventBool.....Pag. 43

Taula 12: Crida d'un RegisterEventl tipus EVREventBool.....Pag. 44

Taula 13: Escena de testeig.....Pag. 45

Taula 14: Cubs amb DraggableInteractiveItem + GrabInteractiveItem.....Pag. 46

Taula 15: Botons TimeBasedIntteractiveItem i SimpleClickInteractiveItem.....Pag. 47

Taula 16: Cub amb un TouchableInteractiveItem.....Pag. 47

Taula 17: Cub amb un GrabInteractiveItem.....Pag. 48

Taula 18: InputManager de Unity.....Pag. 50

Taula 19: Exemple de botons R_Trigger i L_Trigger del InputManaager.....Pag. 51

Taula 20: Disposició de botons HTC VIVE.....Pag. 51

Taula 21: Disposició de botons Oculus Rift.....Pag. 53

Taula 22: Perfil d'usari de la Asset Store Publisher.....Pag. 58

Taula 22: Perfil d'usari de la Asset Store Publisher.....Pag. 59

Índex de figures

Glossari

Unity: Unity és un motor de joc multiplataforma desenvolupat per Unity Technologies, el primer anunci i llançat al juny de 2005 a Worldwide Developers Conference d'Apple Inc. com a motor de joc exclusiu per a OS X. A partir del 2018, el motor s'ha ampliat per donar suport a 27 plataformes. El motor es pot utilitzar per crear jocs tridimensionals i bidimensionals, així com simulacions per a les seves múltiples plataformes. Algunes de les principals versions d'Unity s'han publicat des del seu llançament, sent la darrera versió estable Unity 2018.3.8.

Asset: Qualsevol suport o dades que es puguin utilitzar en el vostre joc o projecte. Un actiu pot provenir d'un fitxer creat fora d'Unity, com ara un model 3D, un fitxer d'àudio o una imatge. També podeu crear alguns tipus d'actius a Unity, com ara un controlador d'animació, un àudio o una textura.

Asset Store: Una biblioteca creixent d'assets comercials gratuïts o de pagament creats per Unity i membres de la comunitat. Ofereix una gran varietat d'assets, des de textures, models i animacions fins a exemples de tutorials sencers, tutorials i extensions d'editor.

VR: Acrònim de Realitat virtual.

C#: (pronunciat C sharp) és un llenguatge de programació multi-paradigma de propòsit general que engloba disciplines de programació orientades a objectes (basades en classes) i orientades a components. Va ser desenvolupat al voltant del 2000 per Microsoft dins de la seva iniciativa .NET i posteriorment aprovat com a estàndard per Ecma (ECMA-334) i ISO (ISO / IEC 23270: 2018). C # és un dels llenguatges de programació dissenyats per a la Common Language Infrastructure.

UI: La interfície d'usuari (IU), en el camp del disseny industrial de la interacció home-ordinador, és l'espai on es produeixen les interaccions entre humans i màquines. L'objectiu d'aquesta interacció és permetre el funcionament i el control efectius de la màquina des del final humà, mentre que la màquina proporciona informació simultània que facilita el procés de presa de decisions dels operadors.

Git: És un sistema de control de versions distribuït per al seguiment dels canvis en el codi font durant el desenvolupament de programari. Està dissenyat per coordinar el treball entre els programadors, però es pot utilitzar per fer el seguiment dels canvis en qualsevol conjunt de fitxers. Els seus objectius inclouen la velocitat, la integritat de les dades i el suport per a fluxos de treball no lineals distribuïts.

Github: És un servei d'allotjament basat en web per controlar versions amb Git. S'utilitza principalment per a codis informàtics. Ofereix totes les funcions de control de versions distribuïdes i de gestió de codi font (SCM) de Git i també afegeix les seves pròpies característiques.

Multi-plataforma: És un terme informàtic que s'utilitza per definir al programari, ja sigui un sistema operatiu, llenguatge de programació, programa, etc. que pot ésser executat en diverses plataformes. Una aplicació multiplataforma pot executar-se en totes les plataformes més comunes o simplement en més d'una. Per exemple una aplicació multiplataforma seria capaç d'executar-se a Windows, Linux i Mac OS X, ja sigui en un PowerPC o un X86.

SteamVR: Valve manté aquest asset de Unity per intercanviar dades sense problemes amb SteamVR. Amb SteamVR, els desenvolupadors poden orientar una API a la qual es poden connectar tots els populars dispositius de PC que implementen realitat virtual. El modern SteamVR Unity Plugin gestiona tres coses principals per als desenvolupadors:

- carregar models 3D per a controladors de VR,
- gestionar les entrades d'aquests controladors
- estimar el que sembla la vostra mà mentre utilitzeu aquests controladors.

Proporcionar exemples concrets d'interacció amb el món virtual i les nostres API.

Oculus: The Oculus Integration brings advanced rendering, social, platform, audio, and Avatars development support for Oculus VR devices and some Open VR supported devices.

Monobehaviour: MonoBehaviour és la classe base a partir de la qual deriva cada script Unity. Quan utilitzeu C #, heu de derivar explícitament de MonoBehaviour.

Aquesta classe no suporta l'operador condicional nul i l'operador de coalescència nul. Hi ha una casella de selecció per desactivar MonoBehaviour a l'editor Unity. Desactiva les funcions quan no es selecciona. Si cap d'aquestes funcions està present en

l'escriptura, l'editor no mostrarà la casella de selecció. Les funcions són: Start(), Update(), FixedUpdate(), LateUpdate(), OnGUI(), OnDisable() i OnEnable().

Prefab: El sistema Prefab de Unity us permet crear, configurar i emmagatzemar un GameObject amb tots els seus components, valors de propietat i GameObjects fills com un actiu reutilitzable.

L'actiu prefabricat actua com a plantilla des del qual podeu crear noves instàncies prefabricades a l'escena.

Quan vulgueu reutilitzar un GameObject configurat d'una manera determinada: com un personatge que no sigui un jugador (NPC), un prop o un escenari en diversos llocs de la vostra escena o en diverses escenes del vostre projecte, haureu de convertir-lo en un Prefab. Això és millor que simplement copiar i enganxar GameObject, ja que el sistema Prefab us permet mantenir automàticament totes les còpies.

Les edicions que realitzeu a un actiu prefabricat es reflecteixen automàticament en les instàncies d'aquest prefab, que us permetran fer grans canvis a tot el projecte sense haver de repetir la mateixa modificació a totes les còpies de l'actiu.

Podeu crear nested prefabs dins d'altres prefabs per crear jerarquies complexes d'objectes fàcils d'editar a diversos nivells.

Tanmateix, això no vol dir que totes les instàncies de prefabs han de ser idèntiques. Podeu anul·lar la configuració de les instàncies de prefabs individualment si voleu que algunes instàncies d'un prefab sigui diferent de les altres. També podeu crear variants de prefabs que us permetran agrupar un conjunt de substitucions en una variació significativa d'un prefab.

També haureu d'utilitzar prefabricacions quan vulgueu crear instàncies de GameObjects en temps d'execució que no existien a l'escena al principi. Per exemple, per fer accions, efectes especials, projectils o NPCs apareixeran als moments adequats durant el joc.

Alguns exemples comuns d'ús de prefabs són:

- Actius ambientals: per exemple, un determinat tipus d'arbre utilitzat diverses vegades al voltant d'un nivell.

- Personatges que no siguin jugadors (NPC): per exemple, un determinat tipus de robot pot aparèixer en el joc diverses vegades, a través de diversos nivells. Poden ser diferents (utilitzant les substitucions) en la velocitat que mouen o en el so que fan.

- Projectils: per exemple, el canó d'un pirata podria crear una bola de canó prefabricada cada vegada que es dispara.

- El personatge principal del jugador: el jugador prefabricat es pot col·locar al punt de partida de cada nivell (escenes separades) del joc.

Frame: Un frame està compost

Interface: Una interfície només conté les signatures de mètodes, propietats, esdeveniments o indexadors. Una classe o estructura que implementa la interfície ha d'implementar els membres de la interfície especificats a la definició de la interfície.

Wrapper: El terme "Wrapper" s'envia molt a prop. Generalment s'utilitza per descriure una classe que conté una instància d'una altra classe, però que no exposa directament aquesta instància. L'objectiu principal de l'embolcall és proporcionar una manera "diferent" d'utilitzar l'objecte envoltat (potser l'embolcall proporciona una interfície més senzilla o afegeix algunes funcionalitats).

La paraula "wrapper" també es pot utilitzar quan es descriuen patrons de disseny clàssics.

Ajustar un objecte per proporcionar-li una interfície simplificada sovint es descriu com el patró "Facade". El wrapper és el facade.

De vegades és possible que es tingui una classe adequada a una interfície específica, però no es pot canviar el codi perquè sigui conforme a aquesta interfície. Es pot crear un wrapper per a aquesta classe que implementi la interfície, però que dirigeix la majoria de les trucades a l'objecte envoltat. Aquest és el patró Adapter. El wrapper és l'adapter.

La instància que es descriu, on es té dues classes que poden fer la classificació de matrius usant diferents algorismes, sona com el patró 'Estratègia', on us proporcionarà una manera de realitzar una operació en algun objecte, però l'algorisme utilitzat per a aquesta operació pot ser diferent sobre l'estructura d'aquest objecte.

La gran majoria dels embolcalls existeixen per ocultar algun tipus de complexitat.

Instance: Una instance, en C #, és una variable de qualsevol tipus continguda en una classe o estructura i s'utilitza per emmagatzemar dades d'objectes. És membre del seu tipus que conté amb una còpia del camp per a cada instància del tipus que conté.

Els Instance Fields representen les dades d'una classe que permet a un objecte mantenir el seu estat. Aquests camps solen exposar-se com a propietat mitjançant la qual es pot canviar la implementació interna del camp segons les millores en el disseny de la classe sense introduir cap canvi de ruptura. Aquest avantatge nega l'efecte de la lleugera sobrecàrrega a l'accés als camps mitjançant les propietats.

La intenció principal de dissenyar camps d'instància és encapsular dades que cal accedir a tots els mètodes de la classe i permetre que les dades siguin emmagatzemades durant tota la vida de la instància de la classe. A més, es poden evitar les dades de la corrupció accidental ocultant-lo amb el nivell d'accessibilitat requeri. El camp d'instància també es coneix com a variable d'instància.

ScriptableObject: Una classe de la qual podeu derivar si es vol crear objectes que no necessiten estar connectats als objectes del joc. Això és molt útil per a actius que només tenen l'objectiu d'emmagatzemar dades. Aquesta classe no suporta l'operador condicional nul i l'operador de coalescència nul.

Flag: Terme que descriu que un objecte està marcat per a ser tingut en conte en accions posteriors. En programació, s'equipara sovint un flag a un booleà.

String: Les strings són objectes que representen seqüències de caràcters. La classe de cadena estàndard proporciona suport per a aquests objectes amb una interfície similar a la d'un contenidor estàndard de bytes, però afegint funcions dissenyades específicament per operar amb cadenes de caràcters de byte únic.

Classe abstracta: Les classes abstractes són similars a les interfícies. No els podeu instanciar, i poden contenir una barreja de mètodes declarats amb o sense implementació. Tanmateix, amb classes abstractes, podeu declarar camps no estàtics i finals i definir mètodes concrets públics, protegits i privats.

SerializeField: Força la unitat per serialitzar un camp privat. Quan Unity serialitza els vostres scripts, només serialitzarà els camps públics. Si, a més, també voleu que Unity serialitzi un dels vostres camps privats, podeu afegir l'atribut SerializeField al camp.

Unity serialitzarà tots els components del vostre script, recarregarà els nous muntatges i recreará els components del vostre script des de les versions serialitzades. Aquesta serialització no passa amb la funcionalitat de serialització de .NET, però amb un Unity intern.

El sistema de serialització utilitzat pot fer el següent:

- Pot serialitzar els camps públics no estàtics (de tipus serialitzables)

- Pot serialitzar camps no estàtics no públics marcats amb l'atribut SerializeField.

- No pot serialitzar els camps estàtics.

- No es pot serialitzar propietats.

El camp només es serialitzarà si és d'un tipus que Unity pot serialitzar Els tipus de sèrie són:

- Totes les classes que pertanyen a UnityEngine.Object, per exemple GameObject, Component, MonoBehaviour, Texture2D, AnimationClip.

- Tots els tipus de dades bàsics com int, string, float, bool.

- Alguns tipus integrats com Vector2, Vector3, Vector4, Quaternion, Matrix4x4, Color, Rect, LayerMask.

- Matrius de tipus serialitzable

- Llista d'un tipus serialitzable)

- Enums

- Estructures

Raycast: Injecta un raig, des del punt d'origen, en la direcció de la direcció, de la longitud màx. Distància, contra tots els col·lisionadors de l'escena.

Opcionalment, podeu proporcionar un LayerMask, per filtrar qualsevol col·lisionador amb el qual no interessi generar col·lisions.

Rigidbody: Control de la posició d'un objecte a través de la simulació física. Afegir un component Rigidbody a un objecte posarà el seu moviment sota el control del motor físic de Unity. Fins i tot sense afegir cap codi, un objecte Rigidbody serà tirat cap avall per la gravetat i reaccionarà a les col·lisions amb objectes entrants si el component Collider correcte també és present.

El Rigidbody també té una API de seqüència de comandaments que us permet aplicar forces a l'objecte i controlar-la de manera física i realista. Per exemple, el comportament d'un cotxe es pot especificar en funció de les forces aplicades per les rodes. Atesa aquesta informació, el motor de física pot gestionar la majoria dels

aspectes del moviment del cotxe, de manera que s'accelerarà de manera realista i respondrà correctament a les col·lisions.

Joint: Podeu adjuntar un objecte del tipus RigidBody a un altre o a un punt fix de l'espai mitjançant un component Joint. En general, es vol que una articulació permeti almenys certa llibertat de moviment i, per tant, Unity proporciona diferents components de la unió que imposen diferents restriccions.

Per exemple,un Hinge Joint permet la rotació al voltant d'un punt i un eix específics mentre que un Spring Joint manté els objectes separats, però deixa que la distància entre ells s'estengui lleugerament.

BoxCollider: Un col·lisionador primitiu en forma de caixa.

Collider: Els components Collider defineixen la forma d'un objecte per a col·lisions físiques. Un collider, que és invisible, no necessita tenir la mateixa forma que la malla de l'objecte i, de fet, una aproximació aproximada sovint és més eficient i indistinguible en el joc.

Editor de Unity:

Callback: Un callback és un tipus que representa referències a mètodes amb una determinada llista de paràmetres i un tipus de retorn. Quan s'instal·la un delegat, podeu associar la seva instància a qualsevol mètode amb una signatura i un tipus de retorn compatibles. Podeu invocar (o trucar) el mètode a través de la instància delegada.

Els callbacks s'utilitzen per passar mètodes com a arguments a altres mètodes. Els controladors d'esdeveniments no són més que mètodes que s'invoquen a través de callbacks. Creeu un mètode personalitzat i una classe com ara un control de Windows pot trucar al vostre mètode quan es produeix un esdeveniment determinat.

Getter: És aquell mètode que retorna el valor d'una variable de la classe membre privada.

Setter: És aquell mètode que assigna un valor a una variable de la classe membre privada.

Dictionary: El tipus Dicctionary representa una col·lecció de claus i valors de parell de dades. La classe Dicctionary de C #està definida a l'espai de nom

System.Collections.Generic és una classe genèrica i pot emmagatzemar qualsevol tipus de dades en forma de claus i valors. Cada clau ha de ser única a la col·lecció.

Mapping : En el sentit més general, el mapping en la programació significa prendre diverses coses i, en certa manera, associar-les a una altra cosa.

1. Introducció

1.1 Motivació

La motivació principal és que em dedico al desenvolupament d'aplicacions en realitat virtual i m'he trobat amb el problema que es planteja el qual aquest projecte n'és la solució.

Una altra font de motivació és el poder aprendre a publicar un Asset a la pàgina oficial de Unity i tot el que comporta durant i després de la publicació.

També cal destacar com a motivació el voler aprendre a coses noves i a millorar la manera de programar tan en qualitat com en eficiència.

1.2 Formulació del problema

El problema planteja és el següent. Cada cop hi ha més desenvolupadors interessats en crear aplicacions en realitat virtual. Tot i així, el mercat dels dispositius de realitat virtual és petit però variat.

Per una banda, hi han els dispositius per mòbil. Aquest tipus de dispositiu té unes capacitats més reduïdes i simples i s'utilitzen acompanyats d'un mòbil. Tenen un preu molt assequible (entre 5 i 70 euros) i són molt fàcils de traslladar o moure.

Per una altra, hi han els dispositius per PC. Aquest tipus de dispositiu té una capacitat molt més elevada en comparació amb l'anterior. Per ser utilitzats requereixen un ordinador potent que fa que siguin menys assequibles per a simples aficionats o intermitjos. Ronden entre els 500 i 1200 euros i normalment inclouen un conjunt d'accessoris com per exemple sensors o comandaments. Estan pensats per estar fixes. Degut als grans costos que suposa tenir un dispositiu d'aquest calibre, la gran majoria del públic són empreses o usuaris amb coneixements extensos sobre el tema.

Degut a aquesta diversitat entre diferents dispositius, la manera d'implementar aplicacions també varia segons el dispositiu triat. És per això que desenvolupar una aplicació per dos o més dispositius és complicat i s'han de tenir en compte molts factors.

Actualment, l'Asset Store de Unity no disposa de cap Asset que permeti la multiplataforma per a una mateixa aplicació de manera assequible per a desenvolupadors amateurs o intermitjos i les grans empreses utilitzen la seva pròpia arquitectura per fer-ho. Però, si l'usuari no és una gran empresa o no vol gastar temps

i diners en desenvolupar un mòdul per desenvolupar aplicacions en VR no té una solució fàcil i ràpida d'accedir a un mòdul ja creat.

1.3 Objectius generals del TFG

L'objectiu principal d'aquest projecte és introduir al mercat un Asset de Unity que funcioni com un mòdul que permeti als usuaris desenvolupar aplicacions en VR en multi-plataforma.

Les plataformes incloses són: HTC Vive, Oculus Rift, Samsung Gear VR i qualsevol tipus de Cardboard.

Aquest asset permetrà als desenvolupadors dos funcions principals:

Per una banda, introduir-se en el món del desenvolupament en realitat virtual de manera més fàcil i còmode i pugui experimentar amb les diferents plataformes.

Per una altra banda, desenvolupar aplicacions multiplataforma amb més serietat sense tenir que preocupar-se per la multiplataforma i també aprofitar la arquitectura d'aquest core per tal d'adaptar-la a qualsevol aplicació, és a dir, reaprofitar les bases establertes.

1.4 Objectius específics del TFG

Els objectius específics per aquest projecte són dos:

Per una banda, Ampliar els meus coneixements sobre C# i Unity i també aprendre més sobre les capacitats de la realitat virtual i les possibilitats que existeixen avui dia a l'abast de qualsevol persona.

Per una altra, Aprendre tot el cicle que comporta penjar un Assets al Asset Store de Unity, des del principi, fins després d'haver-se penjat en quant a interacció amb altres usuaris interessats.

1.5 Abast del projecte

Aquest projecte està destinat a convertir-se en un Asset Package penjat al Asset Store. De manera que és un arxiu públic per a tothom que tingui accés al Asset Store de Unity, que es una pagina web oberta a tot el mon.

Per tant, el producte va dirigit a desenvolupadors de Unity interessats en la realitat virtual, especialment persones de nivell principiant o intermitja.

El beneficiat pel resultat del treball serà la pròpia comunitat de desenvolupadors, ja que el arxiu serà totalment gratuït i obert per tal de ser ampliat si es desitja.

2. Estat de l'art

La realitat virtual és una tecnología que en la última dècada ha anat creixent ràpidament, especialment durant aquest últims cinc anys. Però el concepte de realitat virtual va sorgir unes quantes dècades més enrere.

En 1935, Stanley Weinbaum va publicar les Espectacles de Pygmalion: una història de ciència ficció. El personatge principal de la història porta un parell d'ulleres que el transporta a un món de ficció que estimula els seus sentits de manera adequada i que conté enregistraments holográfics. Alguns consideren que és l'origen del concepte de realitat virtual (VR) ja que aquesta història era una bona predicció dels objectius i assoliments del futur. Els primers desenvolupaments tècnics de VR van ser en els anys cinquanta.

Durant els següents anys va experimentar avanços però, es va disparar el creixement a partir del 2014 quan Facebook va comprar l'empresa Oculus VR per $ 2 mil milions. Aquest va ser un moment decisiu en la història de VR perquè VR va guanyar impuls ràpidament després d'això. Sony va anunciar que treballaven en Project Morpheus, un auricular VR per a la PlayStation 4 (PS4). Google va llençar el cartró: un visor estereoscòpic de baix cost i bricolatge per a telèfons intel·ligents. Samsung va anunciar el Samsung Gear VR, un auricular que utilitza un telèfon intel·ligent Samsung Galaxy com a espectador. Més persones van començar a explorar les possibilitats de VR, incloent-hi l'addició d'accessoris innovadors.

Durant els següents anys moltes empreses estan desenvolupant els seus propis auriculars VR, inclosos HTC, Google, Apple, Amazon, Microsoft Sony, Samsung, etc. Sony podria estar desenvolupant una tecnologia de seguiment d'ubicació similar a HTC VIVE per a la PlayStation 4.

Finalment, al 2018 a la Facebook F8, Oculus va demostrar un nou prototip de l'auricular, Half Dome. Es tracta d'un auricular varifocal amb un camp de visió de 140 graus.

La realitat virtual ha progressat significativament i ara s'utilitza de diverses maneres, des de proporcionar experiències de joc immersives, ajudar a tractar trastorns psicològics, a ensenyar noves habilitats i fins i tot a rebre malalts terminals en desplaçaments virtuals. VR té moltes aplicacions i amb l'auge de la tecnologia de smartphones VR serà encara més accessible.

Amb gran nombre d'empreses que competeixen, es controlen nous controladors i molts usos per a VR, aquest camp només pot millorar.

2.1 Estudi de Mercat

Productes que es troben ja en el mercat, semblants al que es proposa en el TFG.

Competència directe:

La competència principal que té aquest projecte és, al ser un producte del Asset Store de Unity, altres assets que tinguin a veure amb la implementació de APIs per implementar realitat virtual en Unity. Dintre d'aquest grup d'assets només n'hi ha una que fa el mateix que el projecte, és a dir, permetre la implementació de realitat virtual per diferents dispositius de manera simple. Tot i que el producte que ofereix l'empresa AVR Works és més complet que el paquet que oferirà aquest projecte, el seu preu és molt més elevat (134 €). Per tant, aquest preu no s'adequa al mercat més principiant i/o intermig que busca ExtendedVR Core.

En quant a la resta d'aplicacions, la gran majoria se centre en un únic dispositiu i implementen un producte molt més senzill a un preu d'entre 5 i 10 €.

Competència indirecte:

La competència indirecte consisteix bàsicament en altres motors de videojocs que permeten desenvolupar aplicacions en realitat virtual.

Per una banda, estan els motors de videojoc gratuïts. L'exponent més rellevant després de Unity és Unreal Engine. Unreal Engine és el màxim competidor de Unity en aquests moments ja que és el segon motor més utilitzat per la comunitat després de Unity. També té un sistema avançat per desenvolupar aplicacions en realitat virtual, tot y que no suporta tant bé les plataformes mòbils. Un altre competidor també és CryEngine. Tot i que és menys utilitzat que els dos anteriors, continua sent una eina molt potent i una competència a tenir en compte.

Per una altra, hi ha un motor de videojocs amb subscripció obligada. Aquest motor rep el nom de AppGameKit VR. Aquest motor és l'únic que està específicament dedicat al desenvolupament d'aplicacions per realitat virtual. Tot i que aquest motor no és molt conegut o si més no, poc utilitzat per la comunitat de programadors.

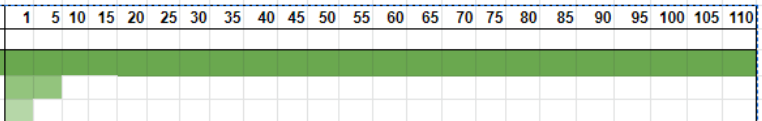
3. Gestió del projecte

3.1 Procediment i Eines per al seguiment del projecte

3.1.1 GANTT

Descripció de totes les fases del desenvolupament en un diagrama de Gantt:

Tasques	Subtasques	Dies
Investigació		
Unity en VR		1-6
Cardboard		2
Oculus Rift		3-4
HTC VIVE		5-6
Prototipació		
Prototip Cardboard		7-12
Prototip Rift		12-18
Prototip VIVE		18-24
Desenvolupament		
Progamació Core		25 - 51
	arquitectura general	25 - 27
	mòdul d'events	27 - 31
	mòdul de Input	31 - 41
	UI	41 - 51
Escnea de Testeig		27 - 61
Correcció d'errors o millores		27 - 110
Testeig		
propi desenvolupador		27 - 110
altres desenvolupadors en VR		65 - 85
altres desenvolupadors en VR		85 - 105
Document TFG		1-110



3.1.2 GitHub repositori en xarxes, Git eines de control de versions

El sistema de repositoris utilitzat és el Git i s'utilitzarà el Github per dur a terme la supervisió del codi. Github també permet la utilització d'un control de versions que també s'utilitza. Finalment, també s'utilitza Github per tenir un control sobre els errors per mitjà del sistema de "Bug reports".

3.2 Eines de validació

Les úniques eines de validació que s'utilitzaran serà una escena de prova o testeig on els usuaris realitzaran tests o simplement utilitzaran l'aplicació per tal de trobar possibles errors o millores.

3.3. DAFO

Els punts forts i dèbils del tema i del seu desenvolupament

	Positius	Negatius
Origen Intern	**Fortaleses** Multi-plataforma Preu molt assequible	**Debilitats** Target amb poc interès en gastar diners
Origen Extern	**Oportunitats** Competència escassa Desenvolupament en VR creixent	**Amenaces** Possibilitat de còpia del codi

3.4. Riscos i pla de contingències

Es de vital importància detectar els riscos que poden posar en perill la feina i buscar solucions per en cas de ser necessari poder reconduir el projecte.

Els possibles riscos identificats d'aquest projecte, i les seves corresponents solucions son les següents, ordenades de menor a major importància:

Risc	Solució
Risc a còpia del projecte	Ús de la llicència MIT com a eina de protecció de dades
Que el projecte no sigui acceptat per a la Asset Store	Seguiment de les guies proporcionades per Unity

3.5. Anàlisi inicial de costos

Aquest projecte no suposa cap cost en el meu cas, ja que el material necessari per desenvolupar i/o testejar me'l facilitarà una empresa.

Per tant, els únics possibles costos que poden haver-hi són:

Per una banda, la compra o llogar el els diferents dispositius (ulleres) necessaris per tal de desenvolupar el projecte. També és necessari disposar d'un ordinador potent, ja que els dispositius de realitat virtual com les Oculus o les VIVE requereixen d'una bona targeta gràfica.

Per una altra, les polítiques de Unity requereixen utilitzar la versió de pagament si el projecte té uns beneficis superiors a x quantitats. Per a més informació consultar l'enllaç de polítiques de pagament de Unity de la bibliografia.

Els costos d'aquest projecte són els següents:

Dispositius	**Preu**
Ordinador	1000-1500€
Oculus Rift	300€
HTC VIVE	600€
Total	**1900-2400€**

Una altra opció és alquilar els dispositius. El preu final varia segons les hores necessàries per desenvolupar el projecte i fer les proves necessàries per tal que funcioni correctament.

4. Metodologia

1. Investigar

a. Estudi de mercat

El primer pas és investigar si hi ha alguna solució al problema plantejat i com implementar la solució en aquest projecte.

En primer lloc, investigar si hi ha algun Asset en la Asset Store algun producte igual o similar al

b. Investigació de les capacitats tecnològiques a l'abast

Investigar les capacitats de les que disposa Unity ara mateix per a crear aplicacions en realitat virtual y com es comporten en diferents dispositius.

c. Investigació de les APIs necessàries per al projecte.

Investigar quines són les APIs necessàries o més convenients per aquest projecte per utilitzar els diferents dispositius.

2. Prototipar

a. Prototipar una prova per a plataformes mòbils

Seguir la guia d'implementació per cardboard de l'enllaç de la bibliografia per provar de crear una senzilla aplicació per cardboard.

Seguir la guia d'implementació per a oculus Gaze del l'enllaç de la bibliografia per provar de crear una senzilla aplicació per a Gaze.

b. Prototipar una prova per a plataformes per a PC

Seguir la guia d'implementació per a oculus Rift de l'enllaç de la bibliografía per provar de crear una senzilla aplicació per cardboard.

Seguir la guia d'implementació per a HTC VIVE de l'enllaç de la bibliografía per provar de crear una senzilla aplicació per cardboard.

c. Prototipar una prova per a multiplataforma

Crear una petita aplicació que funcioni per a un dispositiu per mòbil i un dispositiu per ordinador. L'aplicació ha de ser senzilla, és a dir, simplement que es vegin objectes 3D en VR per als dos dispositius.

3. Desenvolupar

a. Desenvolupament del core

El core que es vol desenvolupar en aquest apartat consisteix en desenvolupar un seguit de classes que controlin diversos sistemes que generin de manera automàtica tots els elements necessaris per tal de que l'usuari pugui implementar la seva pròpia aplicació en realitat virtual amb les mínimes preocupacions possibles.

Aquest conjunt de sistemes inclou:

Un sistema que s'encarrega de generar la càmera especialitzada per a realitat virtual i controlar el input per a cada dispositiu en concret mitjançant les APIs corresponents per a cada dispositiu.

Un sistema que s'encarrega de generar i controlar un sistema d' events que permeten a l'usuari controlar tot el que succeeix en respecte a la interacció d'objectes de manera fàcil i senzilla.

El codi ha d'estar preparat perquè sigui escalable i es puguin implementar nous sistemes per tal d'ampliar el core, per exemple, un sistema de descarrega de dades.

A part dels sistemes o mòduls, el core també implementa un conjunt d'elements de UI. La realitat virtual és una tecnologia nova que elimina qualsevol representació de pantalla o dos dimensions, és a dir., tots els objectes existeixen en l'espai en tres dimensions. Per tant, tota la UI ha de ser representada com a un objecte 3D i això comporta una sèrie de complicacions xafogoses. És per això, que el core també implementa un set de diferents elements de UI per tal de facilitar al usuaris la implementació de menús o altres elements que requereixin UI.

b. Creació d'una escena de mostra

Aquest asset també inclou una escena de testeig la qual posa en pràctica totes les funcionalitat que és capaç de realitzar el core.

4. Testejar

a. Testejar l'escena i les funcionalitats del core

i. El propi desenvolupador

El propi desenvolupadors o desenvolupadors han de dur a terme la primera ronda de testeigs per tal d'intentar eliminar el màxim número de bugs possibles abans de presentar el projecte a usuaris externs. Per tal de trobar i corregir errors s'utilitzarà l'escena de prova per comprovar com es comporta el core en la pràctica.

ii. Usuaris amb experiència en realitat virtual

La segona ronda de testeigs consisteix en que dos usuaris familiaritzats en desenvolupament en Unity i amb realitat virtual testegin l'escena de prova per intentar trobar errors i intentar trobar millores a l'escena

iii. Usuaris sense experiència en realitat virtual

Finalment, la última ronda consisteix en repetir els mateixos passos de la segona ronda però aquest cop amb desenvolupadors familiaritzats amb Unity però no amb realitat virtual.

5. Desenvolupament del projecte

Aparts propis d'aquest tema com pot ser usabilitat, maquetació, disseny, programació, proves, etc.

5.1 Fase 0: Preparació del projecte.

Descarregar el programa Unity Engine. La versió utilitzada per desenvolupar el Extended VR Core és la 2018.3.4f1.

A continuació, descarregar el paquet “SteamVR” de l’Asset Store de Unity. La versió utilitzada per desenvolupar el Extended VR Core és la 2.3.4.

Tot seguit, descarregar el paquet de “” de la pàgina oficial de Oculus. La versió utilitzada per desenvolupar el Extended VR Core és la 6.1.3.

Finalment, crear un projecte nou de Unity i importar els dos paquets anteriors. També caldrà habilitar la Realitat Virtual en el projecte. Això es fa a través de: Project Settings > Player Settings >

Un cop dut a terme tot això, ja es pot començar a desenvolupar el projecte.

Tot el codi és programat amb C#, ja sigui utilitzant les classes heretades de la pròpia API de Unity o creant classes de C# pures.

5.2 Fase 1: Arquitectura de Mòduls

Per tal de mantenir una estructura de codi per l'aplicació, s'ha escollit estructurar el codi per mòduls o sistemes.

La programació modular és una tècnica de disseny del programari que emfasitza separar la funcionalitat d'un programa a mòduls independents, intercanviables, tal que cadascun d'ells conté tot el necessari per a executar només un aspecte de la funcionalitat desitjada.

Una interfície de mòdul expressa els elements que proporciona i requereix el mòdul. Els elements definits en la interfície els poden detectar els altres mòduls. La implementació conté el codi funcional que correspon als elements declarats en la interfície. La programació modular té una relació estreta amb la programació estructurada i la programació orientada a objectes.

Totes tres metodologies, que es varen iniciar al voltant de la dècada dels 60 del segle XX, comparteixen l'objectiu de facilitar la construcció de grans programes i sistemes mitjançant la seva descomposició en partes petites.

Mentre històricament l'ús d'aquests termes ha estat inconsistent, avui en dia la "programació modular" fa referència a la descomposició d'alt nivell del codi d'un programa en peces, programació estructurada a l'ús d'estructures de control de flux a baix nivell i programació orientada a objectes a l'ús d'objectes de dades, una mena d'estructures de dades.

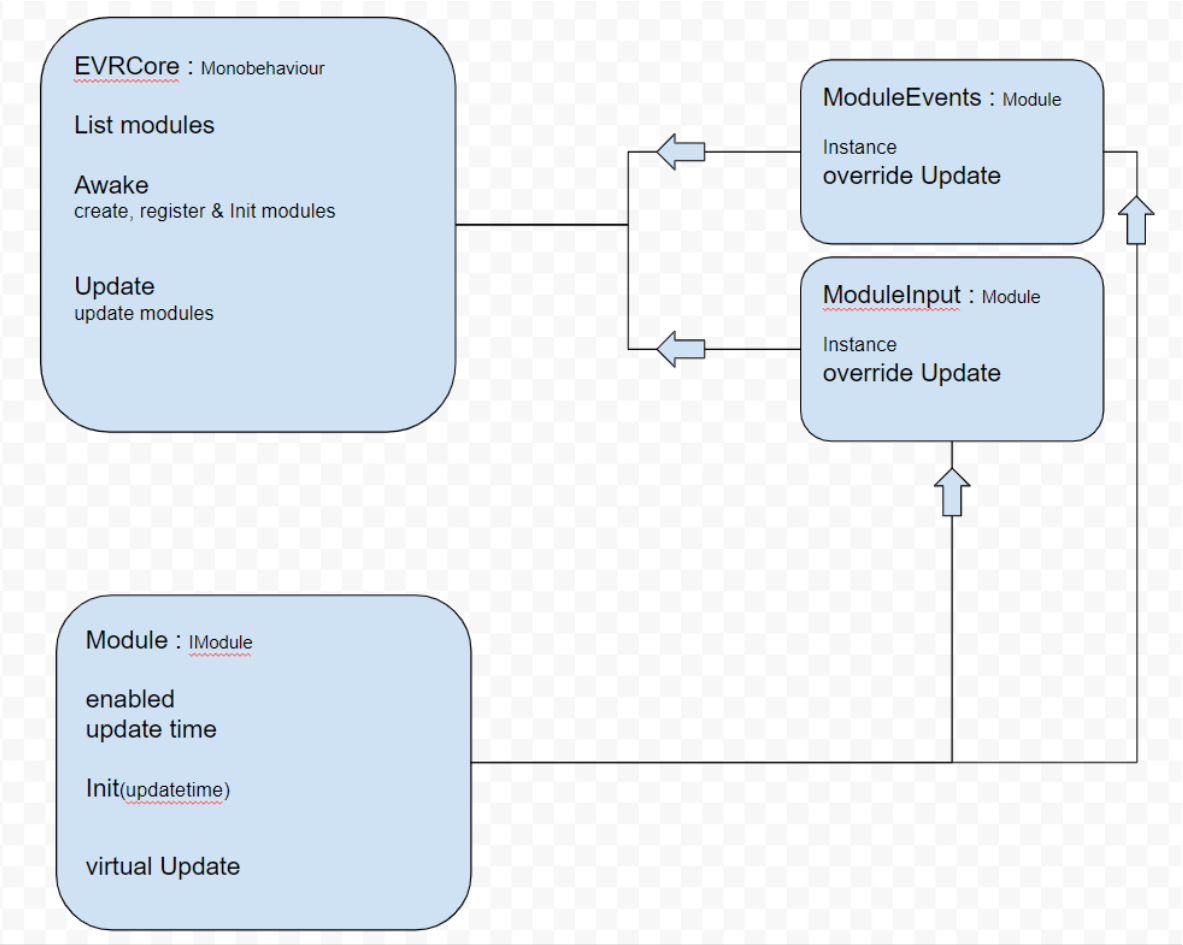
En aquest cas en concret, la classe EVRCore és l'encarregada de crear, inicialitzar, actualitzar i destruir tots els mòduls.

Aquesta classe és una classe que hereta de la classe Monobehaviour de la API de Unity i va vinculada com a component a un Prefab anomenat ExtendedVRCore. Aquest objecte ha d'estar present en qualsevol inici d'aplicació que vulgui utilitzar la lògica del ExtendedVRCore per dur a terme la realitat virtual.

Aquesta classe conté una llista o col·lecció de tots els mòduls necessaris o que es vulguin crear. A la inicialització d'aquesta classe, és a dir, al mètode Awake(), s'han de crear tots els mòduls desitjats i s'han d'afegir a la llista y després s'han de inicialitzar.

Un cop tots els mòduls estan creats, registrats i inicialitzats, s'han d'actualitzar dins del mètode Update(), per tant, cada frame s'ha de recórrer la llista per tal d'actualitzar els mòduls.

També està la opció de crear un comptador personalitzat per actualitzar cada mòdul cada X milisegons. Per determinar el número de milisegons entre iteracions s'ha de passar com a paràmetre a la funció Init() dintre de la classe base mòdul.



Tula 2 – Diagrama UML del sistema d'arquitectura per mòduls del EVRCore.

Cada mòdul ha d'heretar de la classe base Module. Aquesta classe que conté les funcions i les variables bàsiques que ha de tenir qualsevol mòdul. Aquesta classe Module, alhora, hereta d'una Interface anomenada IModule que serveix per especificar quins són els mètodes imprescindibles que ha de tenir qualsevol mòdul.

El codi està pensat per ser escalable, ja que segueix la base d'una arquitectura modular que permet escalabilitat i proporciona independència. Per aquest motiu, l'usuari pot implementar tots els mòduls que requereixi a part dels dos mòduls que ja venen per defecte en el Extended VR Core.

Aquests dos mòduls són els més bàsics per tal de que el core funcioni correctament. Tot i així, hi ha molts més mòduls necessaris per a desenvolupar qualsevol aplicació sigui de realitat virtual o no.

Així que l'usuari, en comptes de crear un nou sistema de mòduls, port reutilitzar aquest sistema ja desenvolupat i ampliar-lo amb nous mòduls com per exemple, un mòdul de descarrega de dades o un mòdul per administrar els recursos o assets.

A continuació s'expliquen els dos mòduls imprescindibles en un Core que s'han desenvolupat.

5.3 Fase 2: ModuleInput

El ModuleInput és el mòdul encarregat de dur a terme tota la interacció entre l'usuari i la aplicació.

Aquest mòdul realitza dos funcions importants:

Per una banda, detectar el dispositiu a qual va destinada la aplicació i generar tota la lògica necessària per tal de vincular el hardware amb la lògica del input, que posteriorment transforma en una lògics comuna per a la resta de l'aplicació.

Per una altra, registrar tots els elements interactius i interpretar el input que genera l'usuari per tal que interactuï amb aquests objectes interactius.

A continuació s'expliquen aquests dos punts més en detall ja que són bastant complexes:

Generació lògica segons el dispositiu:

Tal i com s'ha explicat, el projecte Extended VR Core és core que es basa en la habilitat de permetre desenvolupar aplicacions per a diferents plataformes.

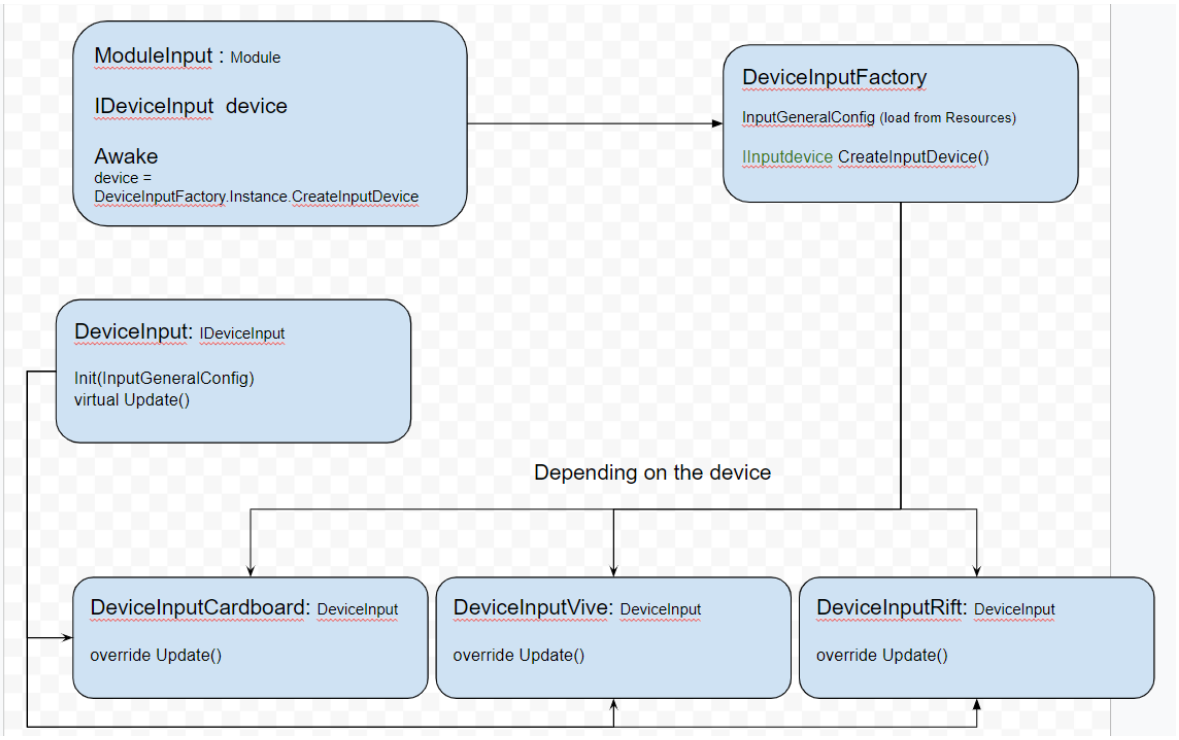
La idea és el ModuleInput detecti el hardware utilitzat pel sistema i transformi la lògica específica per aquest hardware en concret en una segona capa de lògica de input comuna per a tots els hardwares. Aquesta ultima serà la que utilitzaran la resta de classes de l'aplicació per tal de consulta o accedir a qualsevol tipus de Input. És a dir, la classe del ModuleInput actua com a wrapper per a tots els diferents inputs dels diferents hardwares i crea un input global que funciona per a qualsevol tipus de dispositiu.

Per dur a terme aquesta nova capa d'input es realitza la següent estratègia:

Dintre de la inicialització del ModuleInput es crida a una classe anomenada DeviceInputFactory per mitja de la instancia o Instance. Aquesta classe és la encarregada de detectar quin tipus de hardware està connectat o a quin tipus de hardware va destinat l'aplicació en qüestió.

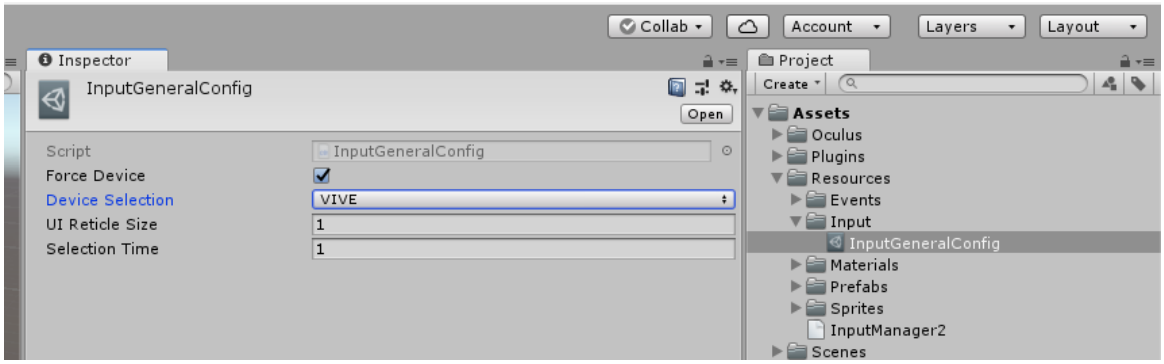
Per cada diferent tipus de dispositiu hi ha una classe especifica. Totes hereten de la classe base DeviceInput i alhora aquesta hereta de la interface IDeviceInput. Segueix la mateixa estructura que els mòdul com es pot veure.

Per tant, hi ha una interface que determina tots els mètodes imprescindibles que ha de tenir qualsevol DeviceInput i una classe DeviceInputBase que hereta d'aquesta interface i determina les variables i els mètodes que han de contenir totes les subclasses del tipus DeviceInput. Aquest classe mare, conte en la seva funció Update tota la lògica que transforma el input del usuari en accions que es duen a terme dins l'aplicació. Però és un mètode virtual i, per tant, cada classe filla ha de aplicar la seva pròpia lògica necessària per administrar el dispositiu en concret a la que va destinada.

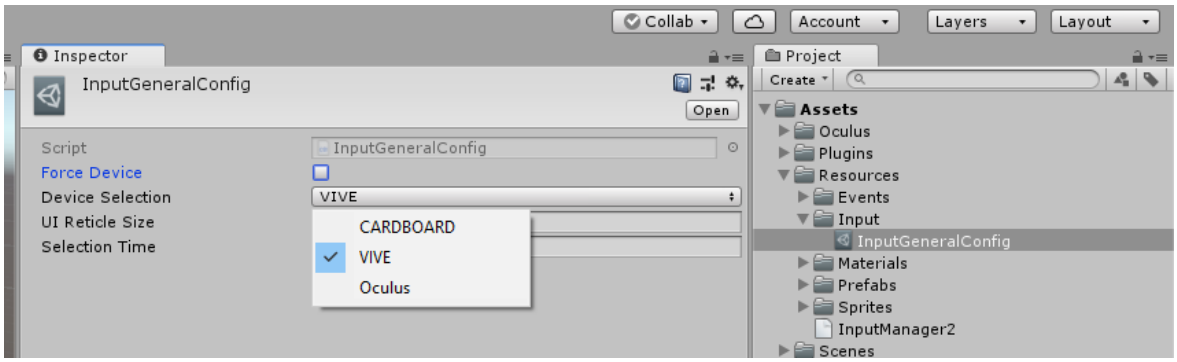


Taula 3: Diagrama UML de Selecció de Hardware

Per tal de determinar el dispositiu, aquesta classe carga dels recursos de Unity un asset anomenat InputGeneralConfig. Aquest asset prové d'un ScriptableObject creat prèviament on es defineixen tots els tipus de dispositius els quals el Extended VR Core permet desenvolupar i un flag per determinar si l'usuari vol forçar el mètode triat o deixar a l'aplicació triar l'opció més recomanable.



Taula 4: Asset InputGeneralConfig.



Taula 5: Asset InputGeneralConfig amb tipus de dispositius disponibles

Un cop aquest asset està carregat, es mira si el flag de forçar està activat o no. Si està activat, es genera una subclasse del DeviceInput específica pel dispositiu escollit.

Si el flag no està activat, la classe DeviceInputFactory utilitza la API de Unity per tal de intentar averiguar el nom complet del dispositiu que està connectat al ordinador. Això

es duu a terme utilitzant la classe XRDevice, accedint al XRDevice.model. Això et retorna una string que es converteix tot a minúscules utilitzant el mètode ToLower() que ve per defecte a la classe string de Unity.

Amb això obtenim el nom del dispositiu connectat. Tot seguit es mira si el nom conté les següents cadenes a dins:

- Si conté la paraula “vive” es genera un DeviceInputVive.
- Si conté la paraula “rift” es genera un DeviceInputRift
- Si no conté cap de les anteriors es genera un DeviceInputCardboard

Al final, la funció CreateDeviceInput() de la classe DeviceInputFactory acaba generant una instancia del tipus IDeviceInput que es queda referenciada en el InputManager.

Aquesta instància de IDeviceInput que conté el module input és actualitzada dintre del la pròpia funció Update(), per tant, el input del dispositiu seleccionat serà actualitzat el mateix número de vegades que el module input. S’ha de tenir en compte alhora de configurar el module input quan es passa com a paràmetre el interval de temps que ha de succeir entre Update i Update, ja que aquest mòdul s’encarrega d’actualitzar qualsevol input que realitza l’usuari. Es recomana que el module input tingui un rati d’actualització (variable updateRate) de 20 milisegons. Per defecte, el core ja inicialitza aquest mòdul amb aquest rati.

Tot això és el que compon la primera funció del mòdul de input però, tal i com indica el propi nom, també s’ha d’encarregar de qualsevol input que realitza l’usuari y de com afecta en el món virtual, és a dir, traduir l’acció de l’usuari amb la reacció en el món fictici.

Aquesta és la segona tasca que duu a terme el aquest mòdul en concret.

El Module Input té dos maneres diferents de interpretar el input generat per l’usuari, és a dir, de traduir el les mateixes accions que fa l’usuari amb dos reaccions diferents.

Per una banda està el sistema d’interactivitat física. Aquest sistema està només disponible per als dispositius per plataforma Windows, és a dir, les HTC VIVE i les Oculus Rift, ja que és necessita un conjunt de comandaments dins d’un espai físic en el món virtual.

Interactivitat física

Aquest sistema permet utilitzar els comandaments de manera que es simula la interacció en el món fictici com si fos el món real. Per exemple, si es vol agafar un objecte, s'ha d'apropar el comandament al objecte i un cop es percebi contacte entre els dos cossos, prémer el botó per tal d'agafar-lo.

La interactivitat física té dos aplicacions bàsiques que permeten a l'usuari interactuar amb els cossos de manera física.

Per una banda, els dos controladors, ja siguin de Rift o de VIVE, contenen un component del tipus script anomenat ControllerGrab. Per ser més específic, tots els components els contindrà el gameobject que representa la mà en l'espai virtual. Aquesta classe Monobehaviour s'encarrega de controlar tots els gameobjects que es poden agafar i si l'usuari interactua amb aquests. Cada controlador pot agafat només un objecte alhora. Per fer-ho, es crea una esfera de col·lisió a la mà que detecta tots els objectes que col·lisionen i que alhora contenen un component GrabableInteractiveItem. S'agafa el primer que col·lisiona i se li aplica un joint al controlador per tal que el gameobject agafat segueixi la posició d'aquest.

Per tal que aquests gameobject reaccionin a les accions de dels comandaments, han de tenir tres components necessaris: el component GrabableInteractiveItem, un BoxCollider (Unity) i un rigidbody (Unity).

La classe GrabableInteractiveItem serveix com a punt de referència per als controladors per detectar si es un gameobject que es pot agafar o no i també per controlar la reacció que pot tenir aquest gameobject al ser agafat per l'usuari.

Per una altra banda, els dos controladors, ja siguin de Rift o de VIVE, contenen també un component del tipus TouchController.Per ser més específic, tots els components els contindrà el gameobject que representa la mà en l'espai virtual. Aquesta classe Monobehaviour s'encarrega habilitar o deshabilitar els dos colliders situats a les dues mans virtuals respectivament.

Respecte als objectes amb els quals s'interactuarà funcionen de la següent manera. Són gameobjects que contenen un component TouchInteractiveItem i un component BoxCollider amb el flag de Trigger activat.

Aquest primer component, s'encarrega de detectar si la caixa de col·lisió del propi objecte col·lisiona amb alguna altra primitiva de col·lisió la qual el gameobject tingui el

tag de “PlayerHands”. Aquest tag només el poden tenir els dos únics gameobjects que generen els controladors virtuals o les mans virtuals. En el cas del dispositiu de VIVE s’anomenen Controller(left) i Controller(Right) i en el cas de les Rift s’anomenen LeftHandAnchor i RightHandAnchor.

És important que els dos objectes independentment de la plataforma també posseeixen un component RigidBody, ja que és necessari per detectar col·lisions, però ha de tenir el flag de Kinematic activat.

A part de detectar les colisons dels controladors/mans de l’usuari, també conté un event del tipus UnityEvent anomenat OnSelect que és llença quan el collider de la mà entra en contacte amb contacte amb l’objecte en qüestió.

Aquest event és del tipus UnityEvent en comptes de EVREvent per tal d’habilitar el sistema d’interfaç que ofereix Unity amb els seus events dintre de l’editor. Això permet més simplicitat i claredat alhora d’afegir funcionalitat en un event. Tot i així, també conté amb mètodes públics per tal de registrar i desregistrar callbacks des de qualsevol punt del codi.

Al final, el resultat és un objecte que al ser apretat com un botó el qual l’usuari pot prémer utilitzant i els controladors i que aquest objecte realitzi qualsevol cosa que es desitgi, només s’ha de subscriure al respectiu event.

Interactivitat per raycast o a distancia

Per una altra banda està el sistema d’interactivitat a distància. Aquest sistema ve per defecte en l’aplicació i esta disponible per tot tipus de dispositius independentment de la plataforma. Aquest sistema d’interactivitat funciona de la següent manera: Cada dispositiu llença un raig o raycast des d’una posició determinada o dos en el cas de tenir dos controladors. Aquest raig serveix com a guia per tal d’indicar al usuari on està apuntant. Per tal de seleccionar un interactive item, cal apuntar al objecte. Cada GameObject que conté un InteractiveItem com a component, ha de tenir un component de tipus collider associat obligatòriament, BoxCollider recomanat. El raig reacciona a les col·lisions amb les diferents BoxColliders que intersecciona, per tant, si col·lisiona amb la BoxCollider del InteractiveItem en qüestió i prem el botó de selecció, aquest InteractiveItem quedarà seleccionat.

En aquest projecte descriurem un interactive item o element interactiu com qualsevol objecte que reaccioni amb el controlador de qualsevol dispositiu.

El sistema de Interactive items del Extended VR Core està desenvolupat de s'ha següent manera:

En primer lloc, es defineix una interfície anomenada IInteractiveItem que determina totes les funcions, públiques i no públiques, que contindrà qualsevol classe que compongui un element interactiu.

Tot seguit, es defineix una classe abstracta que funcionarà com una classe base per a tot tipus d'element interactiu.

Aquesta conté tres parts importants per al funcionament d'un element interactiu:

En primer lloc, una instancia serialitzada (amb l'anotació [SerializeField] de Unity] que descriu les característiques que qualsevol element interactiu. Els atributs que es poden assignar a un InteractiveItem determinat són les següents:

- Autoselect: Fa que un InteractiveItem s'autoseleccioni sempre i quan el raycast del ModuleInput colisioni amb aquest.

- Allow Repeat Selection: Permet a un element interactiu seleccionar-se més d'un cop.

- Allow Unselect: Permet a un element interactiu desseleccionar-se en cas de mantenir-se seleccionat.

- Drag and Drop: Permet a un InteractiveItem adquirir la funcionalitat de mantenir i arrossegar per tal de moure els InteractiveItems.

- Stays Selectet: Permet a un InteractiveItem mantenir-se seleccionat un cop seleccionat.

En segon lloc, crear els events necessaris per tal de controlar les funcionalitats dels diferents InteractiveItems en cadascun dels cinc casos:

- OnEnter: Es llença quan el raycast del ModuleInput colisiona amb la BoxCollider del InteractiveItem en qüestió per primer cop des del últim OnExit.

- OnExit: Es llença quan el raycast deixa de col·isionar amb la BoxCollider del InteractiveItem en qüestió.

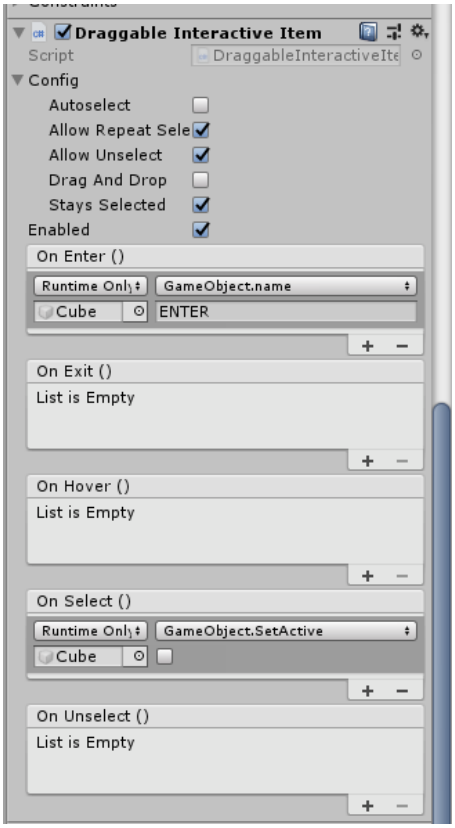
- OnHover: Es llença quan el raycast del ModuleInput es manté colisionant amb la BoxCollider del InteractiveItem en qüestió després d'un OnEnter. Aquest event es llença contínuament sempre que es compleixi la condició descrita anteriorment.

- OnSelect: Es llença quan dins d'un OnHover l'usuari realitza l'acció de seleccionar.

- OnUnselect: Es llença quan un InteractiveItem està seleccionat i l'usuari prem l'opció de desseleccionar.

Aquest events, no són events propis del Core com els descrits posteriorment sinó que són UnityEvent. El motiu pel qual són UnityEvent en comptes de EVREvents perquè, al ser una classe de Unity pròpia, permet una interfície visual més pràctica per a l'usuari dins de l'Editor de Unity.

També s'encarrega de definir les funcions necessàries per registrar callbacks als diferents events.



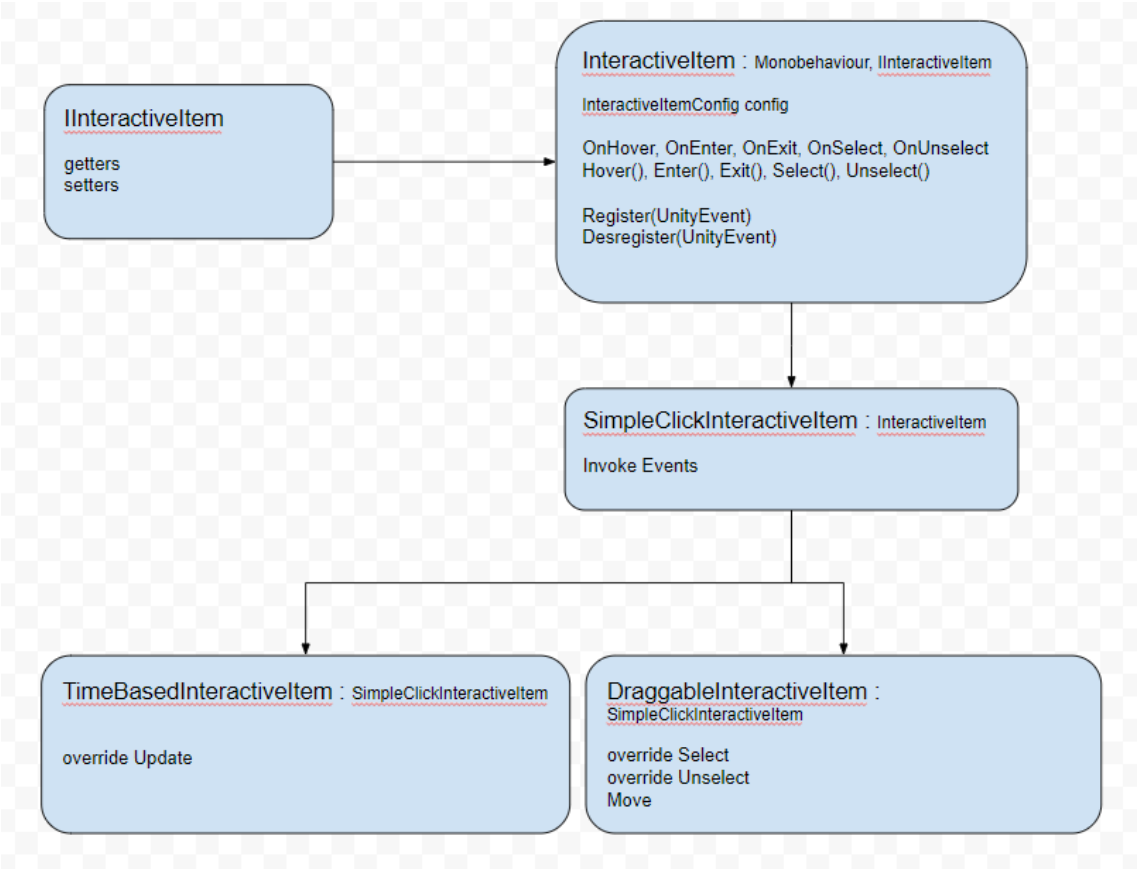
Taula 6 - Configuració InteractiveItem des de l'editor

En tercer lloc, crear els getters i setters necessaris per tal de tenir fàcil accés des de qualsevol punt del sistema de Input per tal de controlar els elements interactius de manera còmode i correcta.

D'aquest classe mare o classe base n'hereta una classe anomenada SimpleClickInteractiveItem. Una classe molt simple que només invoca els mètodes que estan inscrits als events. Aquesta classe permet que un InteractiveItem pugui ser seleccionat i desseleccionat amb les configuracions necessàries. D'aquesta classe en surten dos classes filles més

Per una banda, una classe anomenada DraggableInteractiveItem que permet fer el mateix que la seva classe mare però alhora, afegint la funcionalitat de moure's respecte la posició del raycast del mòdul de Input.

Per una banda, una classe anomenada TimeBasedInteractiveItem que permet fer el mateix que la seva classe mare amb la diferència que els elements interactius se seleccionen automàticament en un temps determinat. Necessiten l'opció d'Autoselect activada.



Taula 7 - Diagrama UML del sistema de InteractiveItems

5.4 Fase 3: ModuleEvents

Event

En informàtica, un event és una acció o una acció reconeguda pel programari, que sovint s'origina de manera asíncrona des de l'entorn extern, que pot ser manejada pel programari.

Com que un event és una entitat que recull l'acció i les variables contextuals que provoquen l'acció, podem utilitzar el mnemònic acròstic d'un event com a "Desencadenant variable d'execució" per aclarir el concepte. Els events informàtics poden generar-se o activar-se pel sistema, per l'usuari o per altres maneres.

Normalment, els events es manegen de manera sincronitzada amb el flux del programa, és a dir, el programari pot tenir un o més llocs dedicats on es gestionin events, sovint un bucle d'events.

Una font d'events inclou l'usuari, que pot interactuar amb el programari mitjançant, per exemple, les pulsacions de teclat al teclat. Una altra font és un dispositiu de maquinari com un temporitzador. El programari també pot activar el seu propi conjunt d'events al bucle d'events, p. Ex. comunicar la realització d'una tasca.

Es diu que el programari que canvia el seu comportament en resposta als events és impulsat per events, sovint amb l'objectiu de ser interactiu.

Sistema d'events

Un sistema d'events és un patró que permet simplificar la manera en com els diferents sistemes o mòduls d'una aplicació o motor es comuniquen entre si eficaçment, mantenint la seva independència pròpia.

Es considera que un sistema d'events té diferents nivells d'implementació que inclouen:

- L'event es rebut per tots els sistemes de l'aplicació
- L'event es rebut per només els sistemes registrats per escoltar events
- L'event es rebut per només els sistemes registrats per escoltar exclusivament aquest tipus d'event
- S'emmagatzemen tots els events fins que són requerits per algun sistema.

Aquest sistema ajuda a mantenir el curs del codi d'execució simple i evitar crear ramificacions inesperades que provoquin efectes col·laterals.

Aquest sistema és recomanable si és necessària una arquitectura de sistemes o mòduls que siguin independents els uns dels altres.

Implementació ModuleEvent

Per crear un sistema d'events, primer és necessari definir una estructura o classe genèrica que permeti contenir qualsevol tipus d'event. Aquesta classe s'anomena EVREvent. Aquesta classe només conté una variable públic del tipus EventArgs.

Aquesta classe, a diferencia d'una classe d'event convencional, hereta de la classe predeterminada de Unity ScriptableObject que permet crear instàncies de tot tipus d'events des de l'editor per tal de facilitar als desenvolupadors o dissenyadors a crear nous events.

Tots els diferents events que es creïn heretan d'aquest classe i per tant mantindran les característiques d'un ScriptableObject i s'encarregaran de sobreescriure la instància de EventArgs.

El ModuleEvent és una classe que hereta de la classe Module explicada anteriorment. S'encarrega de contenir dins d'una col·lecció del tipus dictionary tots els listeners a aquests events. També inclou tres funcions públiques bàsiques per tal que es pugui accedir al sistema d'events des de qualsevol punt del codi:

- RegisterEventListener: Serveix per registrar un objecte a un event determinat. Inclou el tipus d'event i un callback que es cridarà un cop s'hagi fet el "raise" de l'event.

- DesregisterEventListener: Serveix per desregistrar un objecte que ja ha estat prèviament registrat a un event deterrminat. Rep els mateixos paràmetres que l'anterior funció.

- RaiseEvent: Serveix per indicar a un event que ha succeït la condició i, per tant, s'ha d'informar als listeners subscrits a aquest event. Aquesta funció rep tres

paràmetres: L'objecte o instància la qual executa aquest funció, el tipus d'event que es vol llençar i els argument que es necessiten per a llençar quest tipus d'event.

A continuació s'exposa un exemple de com seria la definició d'un tipus d'event el qual retorna un objecte del tipus bool dintre de la variable heretada de la classe EVREvent EnventArgs.

Per poder convertir un booleà en un objecte del tipus EventArgs, és necessari crear una classe que hereti de la clase base de Unity EventArgs. Aquesta classe esta composta per un constructor bàsic, un constructor que rep com a paràmetre un booleà i una variable booleana que determinarà el valor que el propi EventArgs retornarà.

```
public class BoolEventArgs : EventArgs
{
    private bool m_bool;
    public bool Bool
    { get { return m_bool; } private set { } }

    public BoolEventArgs(bool value)
    {
        m_bool = value;
    }

    public BoolEventArgs()
    {
        m_bool = false;
    }
}
```

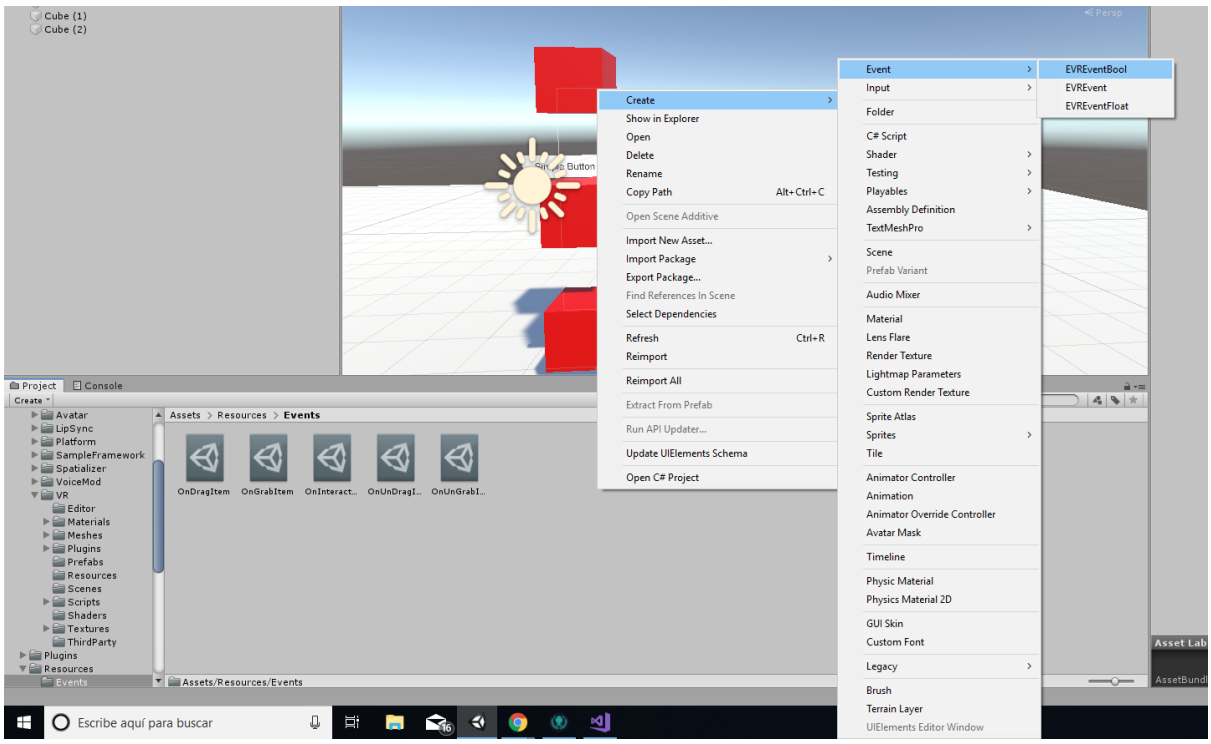
Taula 8 - Exemple d'un EventArgs del tipus booleà

Un cop tenim aquest classe que transforma un booleà amb un EventArgs ara només cal crear una nova classe que hereti de EVREvent i converir la variable heretada EventArgs en una BoolEventArgs.

```
[CreateAssetMenu(fileName = "NewEVREventBool", menuName = "Event/EVREventBool", order = 1)]
public class EVREventBool : EVREvent
{
    public EVREventBool()
    {
        EventArgs = new BoolEventArgs();
    }
}
```

Taula 9 - Exemple d'un EVEREvent del tipus booleà

Un cop aquesta classe ja està definida, ja es poden crear events del tipus EVREventBool, i per tant, events que retornin un booleà a part de l'objecte que ha llençat l'event. Per tal de crear una nova instància d'aquest tipus d'event només cal prémer el botó dret del ratolí a la finestra d'assests i crear una nova variable del tipus EVREventBool.



Taula 10 - Creació d'una Instancia d'event des de l'editor

Un cop realitzat això, apareixerà un asset que representa una instància al ScriptableObject escollit. Per tal d'accedir a aquest asset és necessari realitzar un dels dos passos que hi han a continuació:

- Serialitzar aquest objecte dins d'un component que tingui algun GameObject a l'escena. Això només es pot dur a terme quan el GameObject està a l'escena i sovint no es dóna el cas. Llavors s'ha de seguir la segona opció

- Carregar aquest asset com a un recurs. Per dur això a terme, s'accedeix a la classe Resources i es crida al mètode Load. Aquest mètode rep un template amb el tipus d'objecte que es vol carregar i el path on està ubicat aquest recurs. És **imprescindible que el recurs que es vol carregar estigui dins una carpeta anomenada "Resources"**, ja que la funció Resources.Load només carrega assets que estiguin dins d'una carpeta amb aquest nom. A continuació es mostren dos exemples de com seria subscriure's a un event i com llençar aquest event.

Llença event: El primer paràmetre és l'objecte que llença l'event, en aquest cas és null perquè no és necessari saber-ho. El segon paràmetre és la instància de l'event desitjat, en aquest cas és un event del tipus EVREventBool. Finalment, l'últim paràmetre és el EventArg, en el cas que s'hagi d'enviar algun paràmetre extra com en aquest en particular on s'envia un BoolEventArgs que rep com a paràmetre un booleà.

```
ModuleEvents.Instance.RaiseEvent(null, Resources.Load<EVREventBool>("Events/OnInteractionModeChanged"), new BoolEventArgs(m_dragInputSystem));
```

Taula 11 - Crida d'un RaiseEvent del tipus EVREventBool

Subscriure event: El primer paràmetre és la instància de l'event desitjat, en aquest cas és un event del tipus EVREventBool. El segon paràmetre és la funció que servirà com a callback un cop aquest mateix event sigui llençat des d'un altre punt del codi. Aquesta funció rep dos paràmetres, un del tipus object que

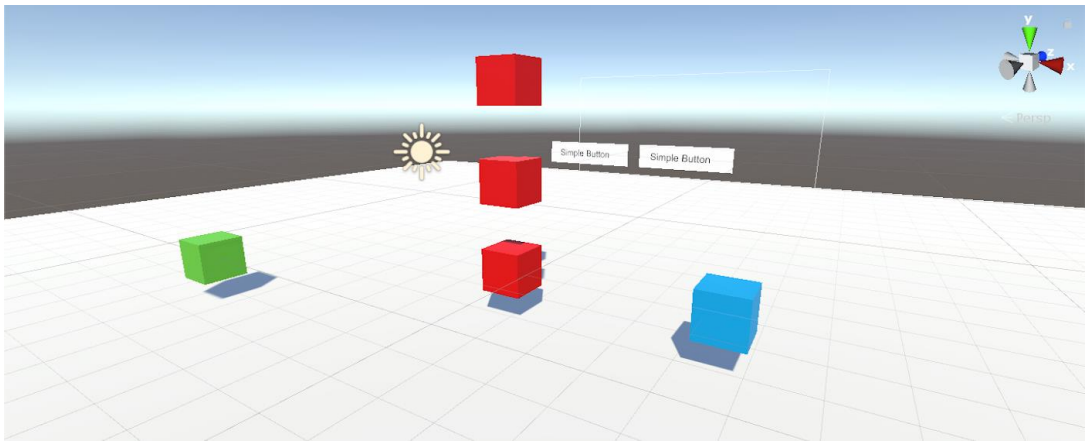
especifica quin és l'objecte que ha llençat l'event i un Eventargs amb els possibles arguments que passa aquest mateix objecte que ha llençat l'event

```
m_enabled = !ModuleInput.Instance.IsDragEnabled();
ModuleEvents.Instance.RegisterEventListener(Resources.Load<EVREventBool>("Events/OnInteractionModeChanged"), OnDragModeChanged);
m_line = GetComponentInChildren<LineRenderer>();
```

Taula 12 - Crida d'un RegisterEvent del tipus EVREventBool

5.5 Fase 4: Test Scene

Aquest fase consisteix en crear una escena de Unity per tal que l'usuari pugui veure les diferents funcionalitats que ofereix el ExtendedVRCore. per tal de recordar les funcionalitats que ofereix aquest asset a continuació escriuré una llista de les diferents features que s'han implementat i de quina manera es poden provar en aquest escena de prova específicament.

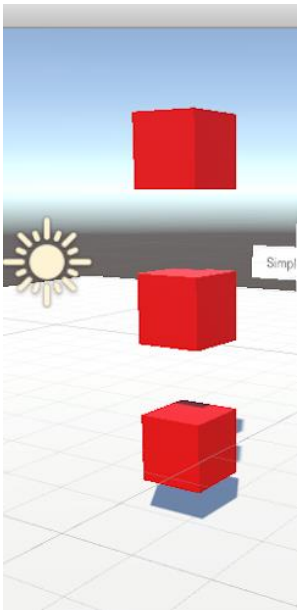


Taula 13 - Escena de testeig

1. Distance Interactive items: Serveixen perquè el usuari pugui interactuar amb ells per mitjà del raig que es llença des del comandament o controlador dret.

   A la escena hi ha diferents elements que contenen un component InteractiveItem, per ser més específics, són sub-elements interactius. Els elements interactius que hi ha a l'escena són els següents:

   - Hi ha tres cubs que contenen cadascun d'ells un component DraggableInteractiveItem que els permet ser seleccionats pel raycast i es mouen en la mateixa direcció que el raig fins que són soltats quan l'usuari acciona el mateix botó de nou.

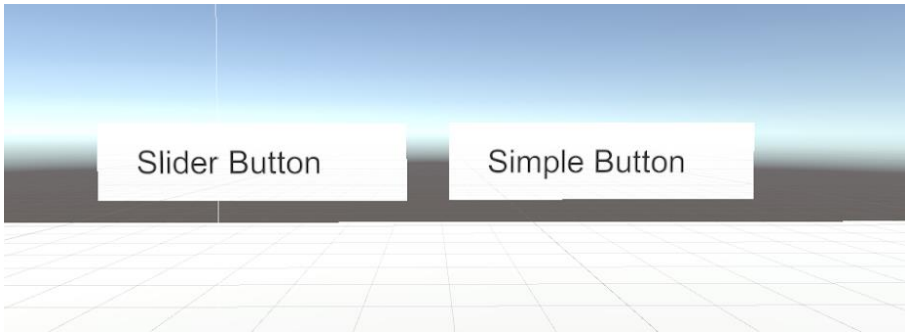


Taula 14 - Cubs amb DragableInteractiveItems + GrabInteractiveItems

- Hi ha un botó que conté un component SimpleClickInteractiveItem que al ser seleccionat per l'usuari, es crearà una esfera al centre de l'escena.

- Hi ha un segon botó a l'escena que conté un component TimeBasedInteractiveItem. Aquest es diferencia respecte l'anterior en el fet que no fa falta prémer cap botó sinó que amb el simple fet d'apuntar a l'element, aquest ja es selecciona automàticament amb un període de temps personalitzable. Durant aquest temps de selecció, es pot percebre com el botó es va omplint de color fins a omplir-se del tot i és llavors quan el botó es selecciona.

  Un cop aquest botó queda seleccionat, es crearà en el centre de l'escena un cub de color blanc.

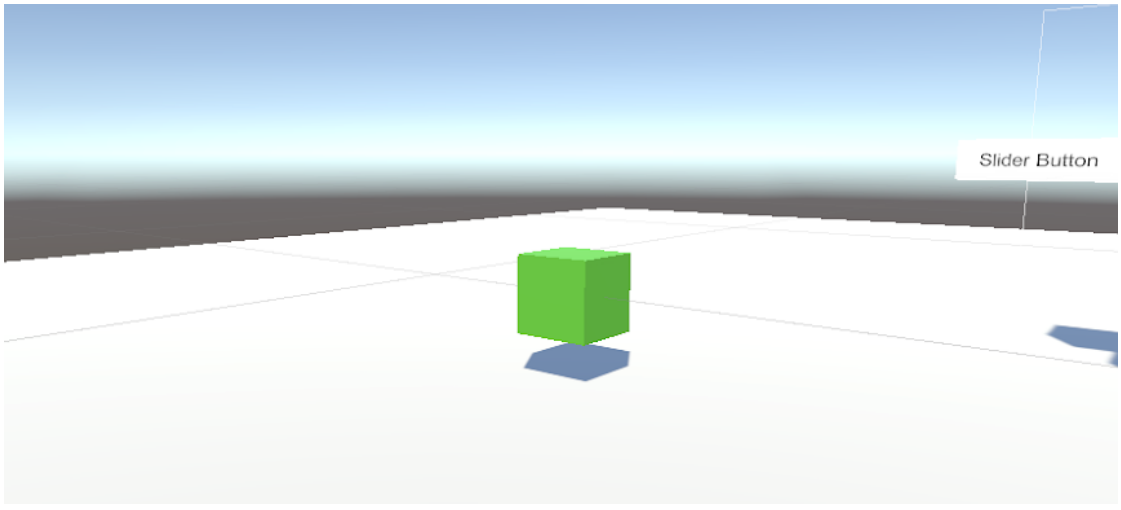


Taula 15 - Botons TimeBasedIntteractiveItem i SimpleClickInteractiveItem

2. Physic Interactive Items: Aquest elements interactius es caracteritzen perquè s'interactua a partir de l'espai físic que ocupa l'usuari que es comparteix entre el món virtual i el món real, és a dir, que se simula l'espai físic en el món virtual.

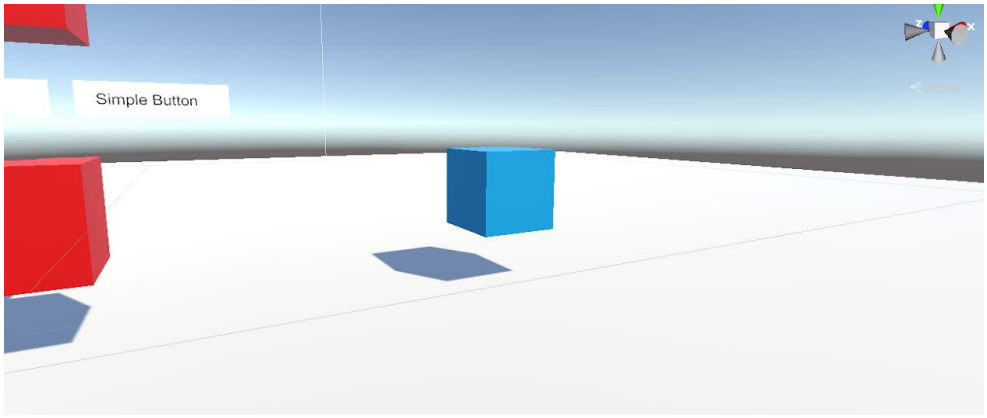
   Hi han dos elements que formen part dins la categoria de Physic Interactive Items:

   - Un cub de color verd que conté un component TouchableInteractiveItem. Un cop aquest cub é seleccionat per l'usuari amb el gest necessari realitzat per l'usuari amb els controladors, es crearà una esfera de color verd en el centre del mapa.



Taula 16 - Cub amb un TouchableInteractiveItem

- Els tres mateixos cubs que s'han comentat en el primer punt, també contenen un component GrabInteractiveItme. Per tant, a part de ser agafats per mitjà d'un raycast, també poden ser agafats per les pròpies mans o controladors virtuals. També hi ha un cub de color blau que conté un component GrabInteractiveItem per interactuar de manera individual



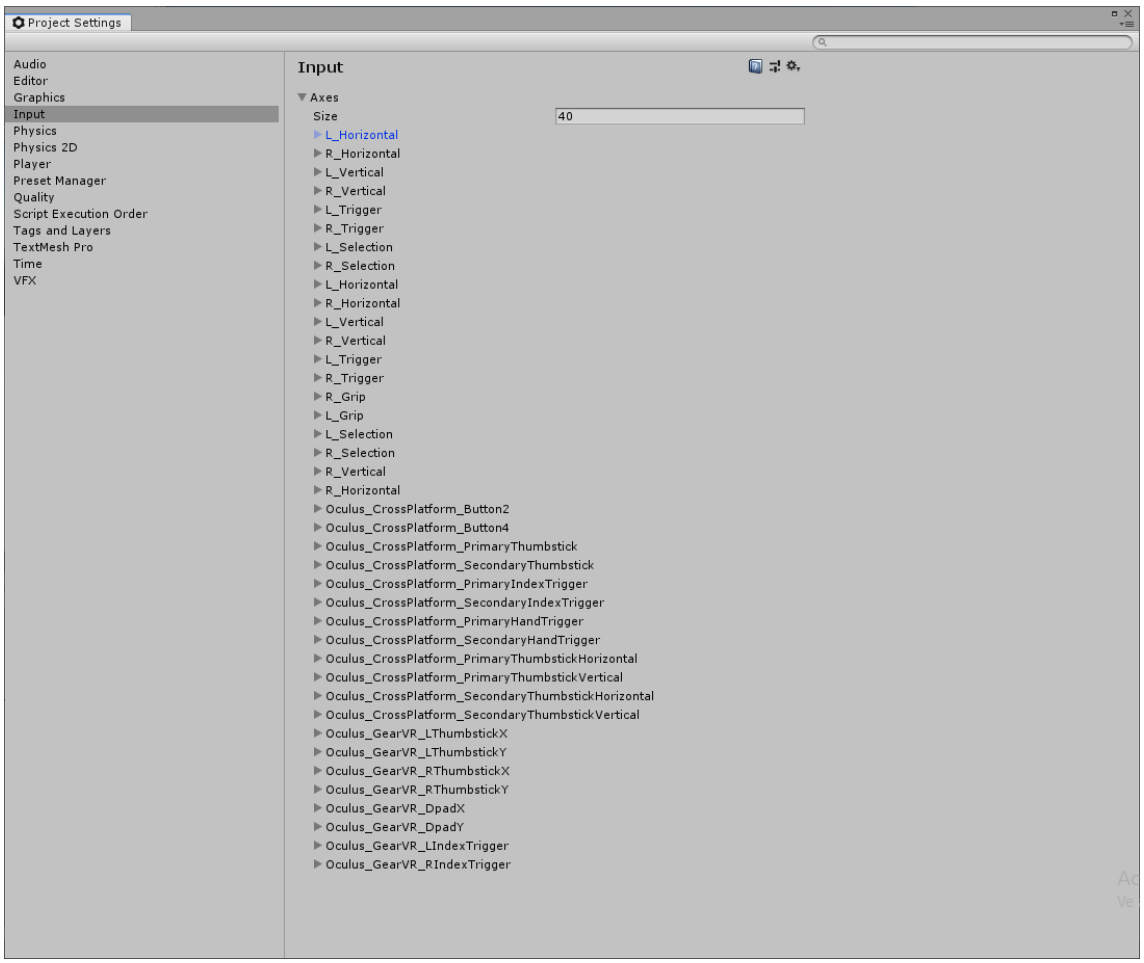
Taula 7 - Cub amb un GrabInteractiveItem

5.6 Fase 5: Controls

En aquest apartat es descriuran els diferents botons que hi han disponibles dins del mapping que ofereix el ExtendedVRCore.

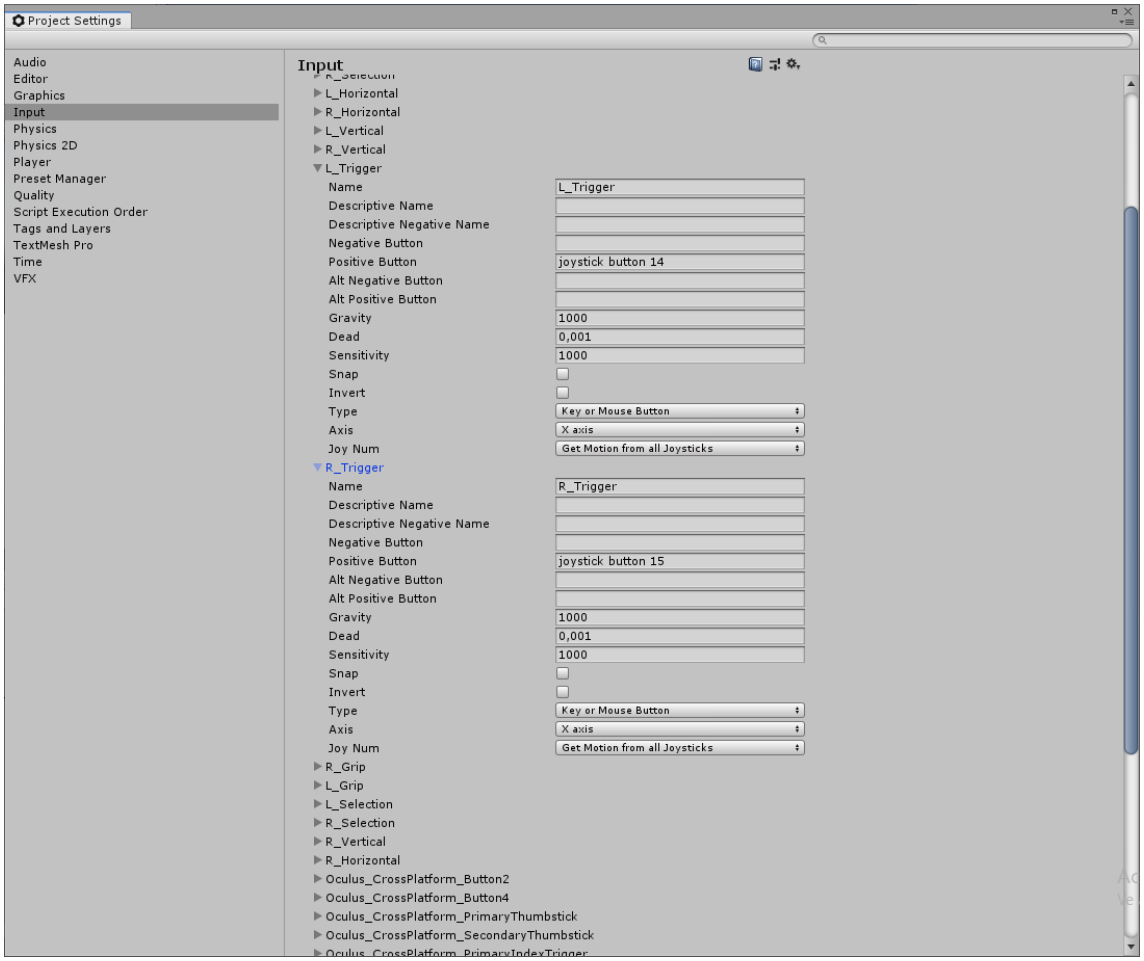
El sistema de input de Unity funciona de manera que cada botó o joystick conté un id. Al Registrar un botó en el InputManager de Unity, el que en realitat es fa és assignar un nom a un id i, és llavors, quan es pot comprovar si aquest botó ha tingut alguna interacció per part del usuari utilitzant la classe predeterminada de Unit Input per mitjà del nom que se li ha assignat anteriorment des de l'editor.

A continuació es mostra una imatge del InputManager que he personalitzat per tal que reconegui amb el mateix nom, diferents IDs, ja que els comandaments dels diferents dispositius no tenen perquè compartir el mateix ID.



Taula 18 – InputManager de Unity

En el cas que es presenta a continuació, l'objectiu és crear el mapping pel botó trigger dels comandaments esquerre i dret per les VIVE. Com que els seus IDs són 14 i 15 respectivament, es crea un botó del tipus L_Trigger i R_Trigger i s'assignen els IDs corresponents.



Taula 19 - Exemple de botons R_Trigger i L_Trigger del InputManaager

Unity utilitza els noms dels diferents botons del InputManager per tal d'obtenir la referencia dels botons en qüestió però, això implica que l'usuari ha d'escriure cada vegada la string corresponent i això pot comportar errors d'escriptura.

Per tal d'evitar possibles confusions es crea una classe estàtica que contindrà variables del tipus string que s'igualen als noms dels diferents botons. Gràcies a això, l'usuari pot accedir a aquestes variables estàtiques públiques per tal d'accedir als botons sense possibilitat d'error en l'escriptura.

Els principals botons que figuren en el mapping són tres que són duplicats per a cadascun dels dos comandaments. Els que contenen la paraula “Main” són pel comandament dret i els que contenen la paraula “Auxiliar” són per l'esquerra. Els tres botons principals són els següents:

- Horizontal_Main_Axis / Horizontal_Auxiliar_Axis & Vertical_MAin_Axi s/ Vertical_Auxiliar_Axis: Els axis Main són els únics axis que també estan disponibles per alguns dispositius del tipus Cardboard, com per exemple, les Samsung Gaze. Els axis Auxiliars només estan disponibles per Oculus Rift i HTC VIVE.

- Main_Selection/Auxiliar_Selection: El botó MainSelection és l’únic botó que també està disponible per alguns dispositius del tipus Cardboard, com per exemple, les Samsung Gaze. El botó AuxiliarSelection només està disponible per Oculus Rift i HTC VIVE.

- Main_Trigger/Auxiliar_Trigger: Aquests dos botons són exclusius de Oculus Rift i HTC VIVE.

- Main_Trigger_Rift: Exclusiu per oculus rift ja que el botó trigger de les oculus rift és pot comportar com un botó o com un axis i per tant conté dos IDs diferents.

- Main_Grip/Auxiliar_Grip: Aquests dos botons són exclusius de Oculus Rift i HTC VIVE.

A continuació es mostren els esquemes dels dos controladors amb comandaments en relació a la ubicació dels botons que contenen. Només es mostren els controladors dels dispositius orientats a PC ja que la majoria de cardboards no contenen hardware.

Taula 20 - Disposició de botons per HTC VIVE

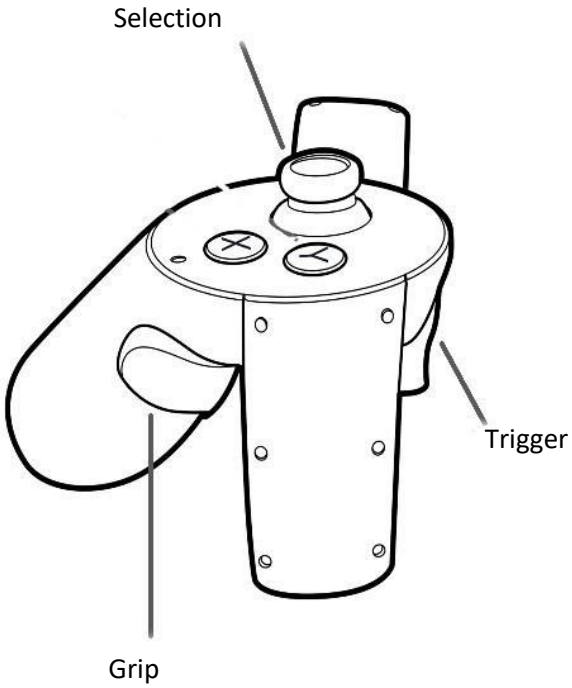


Tabla 21 - Disposició de botons per Oculus Rift

5.7 Fase 6: Testeig i correcció de bugs

Feedback de testejos a 3 persones que entenen de desenvolupament en realitat virtual per Unity:

Segons els tres usuaris que han provat l'aplicació, la idea comuna és que l'escena era una mica massa simple. També s'han corretgit diferents errors o bugs.

Feedback de testeigs a 3 persones que no estan familiaritzades amb el desenvolupament en realitat virtual per Unity però, si en el desenvolupament amb Unity:

Segons els tres usuaris que han provat l'aplicació, la idea comuna és que els controls són complicats per les Rift respecte dels controls per a les VIVE. A vegades els controls de les Rift fallen. També s'han corregit diferents errors o bugs.

Tots els usuaris han entès les diferents funcions d'interactivitat que ofereix l'aplicació i els components que es necessiten per tal de que funcioni el core.

Correcció de bugs i altres errors

- Reconfiguració del Player Settings de Unity per tal de rehabilitar la API de OpenVR que permet que SteamVR funcioni correctament. Es va desactivar temporalment no s'havia tornat a activar de nou.

- El Input de Trigger en els controladors de les Oculus Rift funcionava incorrectament, ja que a vegades es detectava que el botó s'aixecava un cop apretat quant no era el cas, es a dir, en realitat continuava apretat.

  Per solucionar-ho, s'ha decrementat el valor pel qual és determinava quant el botó era soltat o aixecat. Dintre la funció GetTrigger(),de la classe InputDeviceRift, tant per l'esquerre com pel dret s'ha reduït el valor de 0,5 a 0,2.

- El Input de Grip en els controladors de les Oculus Rift funcionava incorrectament, ja que a vegades es detectava que el botó s'aixecava un cop apretat quant no era el cas, es a dir, en realitat continuava apretat.

  Per solucionar-ho, s'ha decrementat el valor pel qual és determinava quant el botó era soltat o aixecat. Dintre la funció GetGrip(), de la classe InputDeviceRift, tant per l'esquerre com pel dret s'ha reduït el valor de 0.5 a 0.2.

- L'acció d'agafar objectes per les Oculus Rift es veia a vegades com s'agafava un objecte amb les mans virtuals no hi havia contacte entre les mans i l'objecte agafat.

  Per Solucionar-ho, al script que hi ha vinculat a cadascun dels anchors d eles mans anomenat GrabControllerGo s'ha canviat la propietat minimumGrabDistance de 0.1 a 0.05.

- En les Oculus Rift, quant s'agafaven dos objectes, un a cada mà, la mà que havia agafat el primer objecte no el deixava anar un cop es deixava d'apretar el botó Grip del controladors.
  Solució, comprovar que si l'altre comandament està agafant un objecte no deshabilitar el control d'aquest des de l'altre mà controlant si la variable currentGrabObject d'una mà és igual al currentObject de l'altre mà.

- En les HTC VIVE, s'ha abaixat la altura a la qual apareix la càmera ja que ja massa alta i no es podia interactuar bé amb els diferents elements de l'escena de testeig.

- En les HTC VIVE, s'ha reduït l'escala de la imatge que és mostra quan s'apunta amb el controlador dret a qualsevol lloc del terra i, per tant, representa la nova ubicació on es teletransportarà l'usuari. S'ha reduït l'escala de l'objecte que conte un component Image de (1, 1, 1) a (0.1, .1, 0.1).

Ampliació de l'escena

Degut a que alguns usuaris van suggerir que l'escena de testeig era molt simple i que estaria bé que s'ensenyés les capacitats que ofereix el ExtendeVRCore.

Per aquest motiu, s'ha ampliat l'escena posant una taula darrere que conté unes quantes pilotes que es poden agafar amb les mans juntament amb una raqueta per tal de poder llençar les pilotes.

Després d'aquest ampliació, els usuaris han donat un feedback més positiu sobre l'escena ja que han pogut experimentar una aplicació real del que seria la interactivitat per mitjà del sistema d'input físic.

5.7 Fase 7: Publicació del projecte

Un cop desenvolupat tot el projecte seguint els passos anteriors ara es hora de crear un paquet i penjar-lo al Asset Store de Unity per tal de que pugui ser adquirit per altres usuaris.

Per dur a terme aquest procés, s'han de realitzar els següents:

(Aquests passos es poden trobar a la guia següent: https://unity3d.com/asset-store/sell-assets)

Crear una conta de Unity

Si es que encara no es disposa d'una conta de Unity, és necessari per tal de poder accedir a les publicacions a la Asset Store.

Crear una conta de Asset Store Publisher

Aquesta cota no té res a veure amb la conta de Unity i és la conta que s'utilitza per publicar paquets a la Asset Store.

Un cop la conta està creada, el següent pas es definir el perfil de usuari. Per fer-ho, s'ha d'anar a la finestra de Info i omplir els diferents camps que es requereixen. També inclou un logo propi amb diferents tamanys.

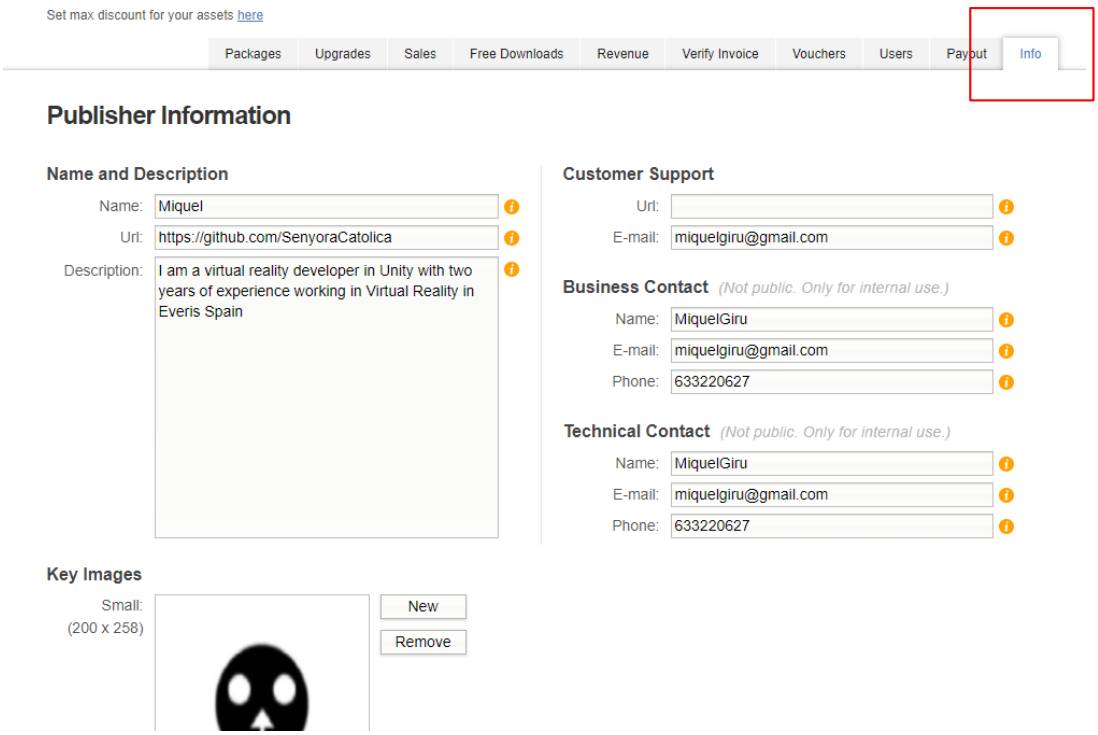


Tabla 22: Perfil d'usuari de l'Asset Store Publisher

En aquest cas, el logo ha estat dissenyat per un company anomenat Aleix Bueso, qui m'ha donat els drets per utilitzar-lo com vulgui.

Crear un nou paquet

Un cop la conta de Asset Store Publisher està creada i definida, és hora de crear el paquet. Unity funciona de manera que qualsevol usuari que vol penjar un paquet primer l'ha d'enviar per tal que el propi personal de Unity revisi el contingut del paquet i accepti que pot ser penjat a la tenda virtual.

El primer pas és crear el prototip del paquet que serà enviat al personal de Unity per a ser revisat. Per dur a terme això és necessari anar a la finestra de Packages i clicar l'opció de Create New Package i omplir tots els requeriments necessaris seguint les instruccions de la pàgina.

Un dels requeriments destacats és el preu que se li aplicarà al producte dins de la tenda virtual. El ExtendedVRCore té un preu de 29.99$ i es consolida com el producte que ofereix realitat virtual amb multiplataforma més barat del mercat a nivell de Asset Store.

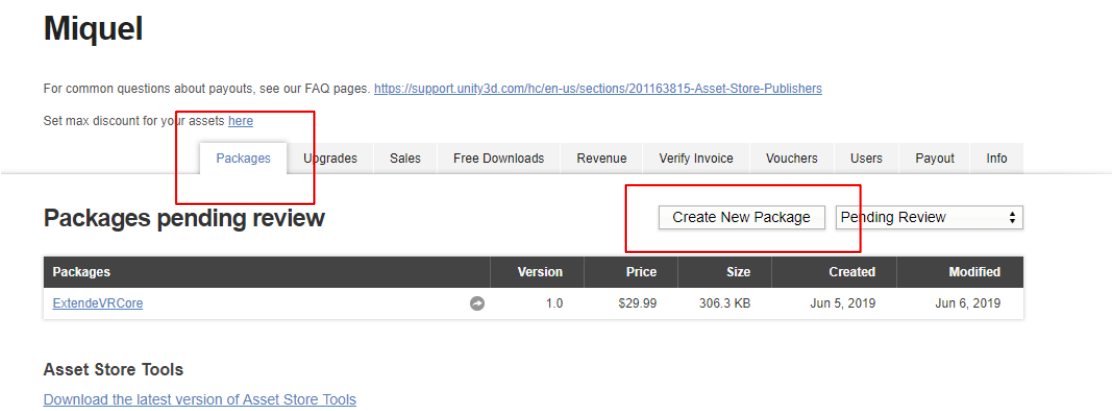


Tabla 23:: Administració de paquets de la Asset Store Publisher

Un cop tota la informació està completa és hora de pujar el projecte vinculat a aquest paquet.

Per fer-ho, és necessari descarregar i importar dins del projecte un Asset del Asset Store anomenat Asset Store Tools. Aquest asset permet crear un paquet per a ser vinculat amb un paquet.

Un cop importat, sortirà una finestra nova a la barra de finestres del editor anomenada Asset Store Tools. El primer pas és seleccionar-la i, posteriorment, seleccionar la primera de les

opcions. A continuació es demanarà l'usuari i la contrasenya de la conta de Publisher i un cop registrats sortiran els paquets disponibles per a editar.

El segon pas és seleccionar el paquet que s'ha creat anteriorment i incloure la carpeta necessària que conté tots els arxius necessaris per tal que l'aplicació funcioni correctament.

Es recomana que abans de realitzar el segon pas, crear una carpeta dins la carpeta d'Assets que contingui tots els arxius necessaris i s'excloguin arxius innecessaris i arxius que no siguin propis, com per exemple la carpeta de Oculus o la de SteamVR.

Finalment se seleccionen els elements que es vol que tinguin vista prèvia en la pàgina web i es prem l'opció de crear el paquet.

Enviament del paquet

Un cop el paquet està penjat, sortirà a la conta de Publisher on anteriorment s'ha creat el paquet. Ja només que acceptar les polítiques d'empresa i altres conceptes i el paquet età llest.

Finalment es clica l'opció final de Submit i automàticament el projecte serà enviat a revisió a Unity i al cap d'uns dies es rebrà la notícia de si ha estat acceptat o no.

En el cas del ExtendedVRCore, ha estat enviat per primer cop el 06/06/2019 i després d'haver sigut revisat per un assistent de Unity el paquet ha sigut denegat per falta d'informació.

En concret, no disposava d'una documentació adequada i la descripció de la pàgina oficial no estava escrita en HTML.

Un cop solucionats aquests errors, s'ha tornat a enviar la petició per penjar el paquet el 19/06/20119. Actualment encara està pendent de revisió.

Nota: Per tal de rebre transaccions dels paquets, també serà necessari vincular la conta de Asset Store Publisher amb una conta de targeta bancaria o de Paypal.

6. Conclusions i treballs futurs

Tots els objectius plantejats a l'inici de del projecte s'han assolit correctament ja que al final s'ha desenvolupat un Package a la Unity Asset Store que permet desenvolupar altres aplicacions per Unity en realitat virtual i ser adaptades amb els mínims canvis possibles fins a tres tipus de dispositius diferents, és a dir, aplicacions per a VIVE, Rift i qualsevol tipus de Cardboard.

Tot el codi es manté obert per als desenvolupadors per tal que sigui possible ser ampliat o fins i tot funcionar com a base per un projecte totalment diferent, ja que s'ha implementat un sistema de mòduls que permet escalabilitat i, alhora, independència entre els diferents mòduls.

Per tant, s'ha aconseguit una nova funcionalitat que en un inici no era l'objectiu del projecte però, si més no, aporta un valor afegit i, per tant, una millora respecte els objectius inicials del projecte.

Conclusions dels dispositius

Cardboard

Aquest dispositiu és el més simple de tots alhora d'implementar, ja que Unity pràcticament ho fa automàticament, només és necessari activar la realitat virtual en els Player Settings.

Aquesta plataforma permet aplicacions extremadament simples ja que la gran majoria de cardboards no disposa pràcticament de cap tipus d'element que permeti al usuari interactuar i enviar algun tipus de input.

Un exemple d'una aplicació que es podria desenvolupar per cardboard seria una experiència visual o algun tipus d'aplicació molt senzilla on s'interactua amb la mirada i amb TimeBasedInteractiveItems.

HTC VIVE

Aquest dispositiu ha suposat una inversió del temps de desenvolupament considerable considerant que gairebé tota la lògica d'interacctivitat amb els diferents tipus de InteractiveItems va estar desenvolupada inicialment per HTC Vive.

Aquest dispositiu ofereix una major qualitat d'imatge i permet crear una àrea de mobilitat més gran que les Oculus Rift

La major debilitat que m'he trobat alhora de desenvolupar diferents funcionalitats és que els controladors de les VIVE són molt simples comparats amb els de les Rift. Una altra de les debilitats és que SteamVR ofereix menys features ja desenvolupades com per exemple un Avatar virtual.

Amb aquest dispositiu es pot desenvolupar qualsevol tipus d'aplicació en realitat virtual per a PC, ja que disposa de totes les vies necessàries per desenvolupar un projecte en VR potent. Això si, es recomana que l'aplicació tingui un sistema d'input simple degut a la simplicitat dels controladors

Oculus Rift

Aquest dispositiu ha sigut el últim en estar implementat hi ha resultat ser el que més còmode d'implementar i el que més possibilitats ofereix per dos simples motius:

En primer lloc, els controladors contenen més botons en comparació a les VIVE i aquest botons contenen doble funcionalitat, ja que serveixen tan per axis com per a botó.

En segon lloc, les funcionalitats o features que ofereix Oculus a través de la seva pàgina web permeten descarregar avatars, models i altres features de manera totalment gratuïta i són molt útils.

Com a inconvenients respecte a les VIVE, ofereixen una menor qualitat d'imatge i de tracking dels sensors. També conté una àrea de joc més reduïda.

Però el que més complicat he trobat alhora d'implementar les Rift al Extended VR Core és el mapping del Input dels controladors. Com he dit anteriorment, els controladors ofereixen moltes possibilitats però això mateix, ha provocat que alhora de controlar els diferents botons hagi hagut de cercar i provar moltes combinacions per a cadascun dels botons. Per tant, com a contrapartida, les Rift tenen un mapping bastant més complicat que les VIVE.

Amb aquest dispositiu es pot desenvolupar qualsevol tipus d'aplicació en realitat virtual per a PC, ja que disposa de totes les vies necessàries per desenvolupar un projecte en VR potent. Tot i així, es recomana que l'aplicació no depengui d'una àrea de joc superior als 3 metres quadrats.

Conclusions sobre Realitat virtual en Unity

En la meva mera opinió, crec que Unity és el motor perfecte per a crear un projecte amb l'objectiu de crear multi plataforma per la simple raó que Unity és el motor de videojoc que més multiplataforma permet a un nivell de qualitat superior.

Unity permet fer tant aplicacions per mòbil com per PC i bàsicament, aquest és l'objectiu principal del Extended VR Core, és a dir, transformar aplicacions en realitat virtual siguin de la plataforma que siguin.

Com a punt negatiu, Unity no ofereix tanta qualitat gràfica en aplicacions per PC com la seva competència directe, és a dir, Unreal Engine. Tot i així, Unreal no permet multi plataforma de tanta qualitat per a mòbil.

És per això que Unity surt com engine que millor s'adapta a les necessitats dels usuaris.

Pel que fa a la implementació de la realitat virtual en Unity, la implementació és molt senzilla i ràpida. Amb un simple click s'habilita la realitat virtual i només falta controlar el input i ja es pot crear una aplicació en realitat virtual simple. El que permet el core, es controlar el input de manera fàcil i còmode gràcies al sistema de mapping de Unity, que permet identificar i assignar els diferents tipus de botons a unes funcionalitats concretes.

En conclusió, Unity és el motor de videojocs més utilitzat en el mercat actualment, especialment per desenvolupadors principiants i intermitjos, i també és el motor més utilitzat per desenvolupar projectes en realitat virtual.

Són per aquests dos motius junt amb el de la simplicitat alhora de desenvolupar en realitat virtual en Unity que he escollit Unity i els resultats han estat molt favorables i amb poques complicacions.

Conclusions sobre la Asset Store

Crear un paquet de Asset per, posteriorment, penjar-lo al núvol de Unity engine és bastant simple ja que Unity conté una guia detallada dels passos que s'han de seguir per tal de penjar un asset a la Asset Store. Tot i així és un procés llarg que pot durar un parell o tres d'hores.

La única cosa que s'ha de tenir en compte alhora de penjar el projecte és:

Per una banda, triar la pròpia llicència del projecte i assegurar-te bé de que protegeixes el teu treball per tal d'evitar possibles problemes de còpies o perdre el reconeixement dintre d'altres projectes.

Per una altra, assegurar-te que el material que s'inclou dins del paquet és teu o tens el permís per penjar-lo.

En el meu cas, jo utilitzo altres paquets de la Asset Store com SteamVR i Oculus Integration. Ja que aquests paquets estan protegits amb llicència, tot i que el seu codi sigui públic, no els puc incloure dins del meu paquet, ja que em podria provocar problemes legals. Per tant, he hagut de incloure en el paquet únicament el material desenvolupat per mi i avisar als usuaris que necessiten altres paquets totalment gratuïts per tal de que l'aplicació funcioni correctament.

També m'ha sorprès la dedicació per part de l'equip de Unity alhora de revisar cada paquet que realitza una petició per a ser pujat a la Asset Store.

M'han demostrat que estan interessats en que els usuaris pengin assets de qualitat i en cas de ser rebutjats, s'informa a l'usuari clarament i punt per punt tots els errors o elements que fallen dins del paquet o de la pàgina del asset en qüestió i de com solucionar aquests error.

Es per això que en aquest aspecte estic molt content amb la dedicació per part de Unity respecte als publishers i a la Asset Store en general.

Objectius futurs

Els objectius futurs pel que fa a aquest projecte són:

En primer lloc, solucionar possibles errors i donar suport i assistència als usuaris que han adquirir el producte i pateixen algun problema o requereixin la meva assistència.

En segon lloc, millorar el codi o expandir les capacitats que ofereixen ara els elements desenvolupats, és a dir, crear una escena més professional, millorar visualment la UI del projecte, adaptar-lo per a dissenyadors o artistes, etc.

En tercer lloc, augmentar la llista de dispositius que suporta actualment el projecte. Cada dia surten nous models de dispositius de realitat virtual i es milloren els existents i, per tant, també s'ha d'actualitzar el ExtendedVRCore per tal d'oferir als usuaris el màxim número de possibilitats i adaptar-se al ritme frenètic en el que s'està desenvolupant la realitat virtual.

Els objectius futurs després d'aquest projecte a nivell personal són seguir desenvolupant amb Unity i C# però en comptes de crear aplicacions en realitat virtual, m'interessa explorar el camp de la realitat augmentada i veure si es pot oferir un producte similar al ExtendedVRCore però centrat en la realitat augmentada. També seria interessant explorar el camp de la realitat mixta.

Tot i així, també vull seguir desenvolupant amb C++, ja que continua sent el llenguatge predominant a les grans empreses desenvolupadores de videojocs on, de moment, espero treballar-hi algun dia.

7. Bibliografia

- Virtual Speech. *https://virtualspeech.com/blog/history-of-vr*. Blog. 12 Febrer, 2019.

- Slant.*https://www.slant.co/topics/2202/~best-game-engines-for-virtual-reality-development*. Blog. 12 Febrer, 2019.

- Asset Store. *https://assetstore.unity.com/search?q=vr&k=vr*. Game Industry Company. 14 Febrer, 2019.

- Cardboard Tutorial. *https://boostlog.io/@mohammedalsayedomar/create-cardboard-apps-in-unity-5ac8f81e47018500491f38c8*. Blog. 16 Febrer, 2019.

- Gaze Tutorial. *https://tutorialsforvr.com/creating-a-gaze-based-ui-for-gear-vr-in-unity/*. Blog. 17 Febrer, 2019.

- Oculus *https://developer.oculus.com/documentation/unity/latest/concepts/book-unity-gsg/*. VR Industry Company. 19 Febrer, 2019.

- Oculus Rift Tutorial. *https://forum.unity.com/threads/unity-oculus-rift-vr-tutorials.488706/*. Blog. 19 Febrer, 2019.

- HTC VIVE.*https://www.vive.com/eu/*. VR Industry Company. 22 Febrer, 2019.

- HTC VIVE Tutorial. *https://www.raywenderlich.com/9189-htc-vive-tutorial-for-unity*. Blog. 22 Febrer, 2019.

- Wikipedia. *https://en.wikipedia.org/wiki/Unity_(game_engine)*. *Encyclopedia*. 5 Febrer, 2019.

- Wikipedia. *https://en.wikipedia.org/wiki/GitHub*. *Encyclopedia*. 5 Febrer, 2019.

- Wikipedia. *https://en.wikipedia.org/wiki/Git*. *Encyclopedia*. 5 Febrer, 2019.

- Wikipedia. *https://en.wikipedia.org/wiki/Git*. *Encyclopedia*. 5 Febrer, 2019.

- Wikipedia. *https://en.wikipedia.org/wiki/User_interface*. *Encyclopedia*. 7 Febrer, 2019.

- AssetStore. *https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647*. Game Industry Company. 15 Març, 2019.

- AssetStore. *https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022*. Game Industry Company. 15 Març, 2019.

- Unity3D Documentation. *https://docs.unity3d.com/ScriptReference/MonoBehaviour.html*. Game Industry Company. 17 Abril, 2019.

- Unity3D Documentation. *https://docs.unity3d.com/Manual/Prefabs.html*. Game Industry Company. 17 Abril, 2019.

- Microsoft Documentation. *https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/interface*. Informatic Industry Company. 19 Abril, 2019.

- StackOverflow. *https://stackoverflow.com/questions/3293752/where-and-how-is-the-term-used-wrapper-in-programming-what-does-it-help-to-do*. Blog. 19 Abril, 2019.

- Technopedia. *https://stackoverflow.com/questions/3293752/where-and-how-is-the-term-used-wrapper-in-programming-what-does-it-help-to-do*. Blog. 24 Abril, 2019.

- Unity3D Documentation. *https://docs.unity3d.com/ScriptReference/ScriptableObject.html*. Game Industry Company. 24 Abril, 2019.

- cplusplus. *http://www.cplusplus.com/reference/string/string/*. Blog. 24 Abril, 2019.

- Oracle Documentation. *https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html*. Informatic Industry Company. 24 Abril, 2019.

- Unity3D Documentation. *https://docs.unity3d.com/ScriptReference/Physics.Raycast.html*. Game Industry Company. 28 Abril, 2019.

- Unity3D Documentation. *https://docs.unity3d.com/ScriptReference/Rigidbody.html*. Game Industry Company. 28 Abril, 2019.

- Unity3D Documentation. *https://docs.unity3d.com/Manual/Joints.html*. Game Industry Company. 28 Abril, 2019.

- Unity3D Documentation. *https://docs.unity3d.com/ScriptReference/BoxCollider.html*. Game Industry Company. 28 Abril, 2019.

- Microsoft Documentation. *https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/index*. Informatic Industry Company. 29 Abril, 2019.

- Sourcecodester. *https://www.sourcecodester.com/book/7682/setter-and-getter-functions-c.html*
. Blog. 29 Abril, 2019.

- C-Sharpcorner. *https://www.c-sharpcorner.com/UploadFile/mahesh/dictionary-in-C-Sharp/*. Blog. 29 Abril, 2019.

- StackOverflow.*https://softwareengineering.stackexchange.com/questions/307639/what-does-mapping-mean-in-programming*. Blog. 19 Abril, 2019.

- Unity3D Documentation. https://unity3d.com/asset-store/sell-assets. Game Industry Company. 3 Juny, 2019.

- Unity3D Documentation. https://assetstore.unity.com/packages/unity/asset-store-tools-115. Game Industry Company. 3 Juny, 2019.